パソコンプログラマのためのクリエイティブマガジン

平成3年4月2日第三種郵便物認可　1991年11月1日発行（毎月1回18日）第1巻第11号通巻11号

月刊**プログラマーズ・ページ**

Programmer's Page

MONTHLY MAGAZINE FOR PROGRAMMERS

1991 11 NOVEMBER

定価980円

特集

Windowsプログラムの作り方

VisualBASIC, TPW, BC++など新処理系登場!!

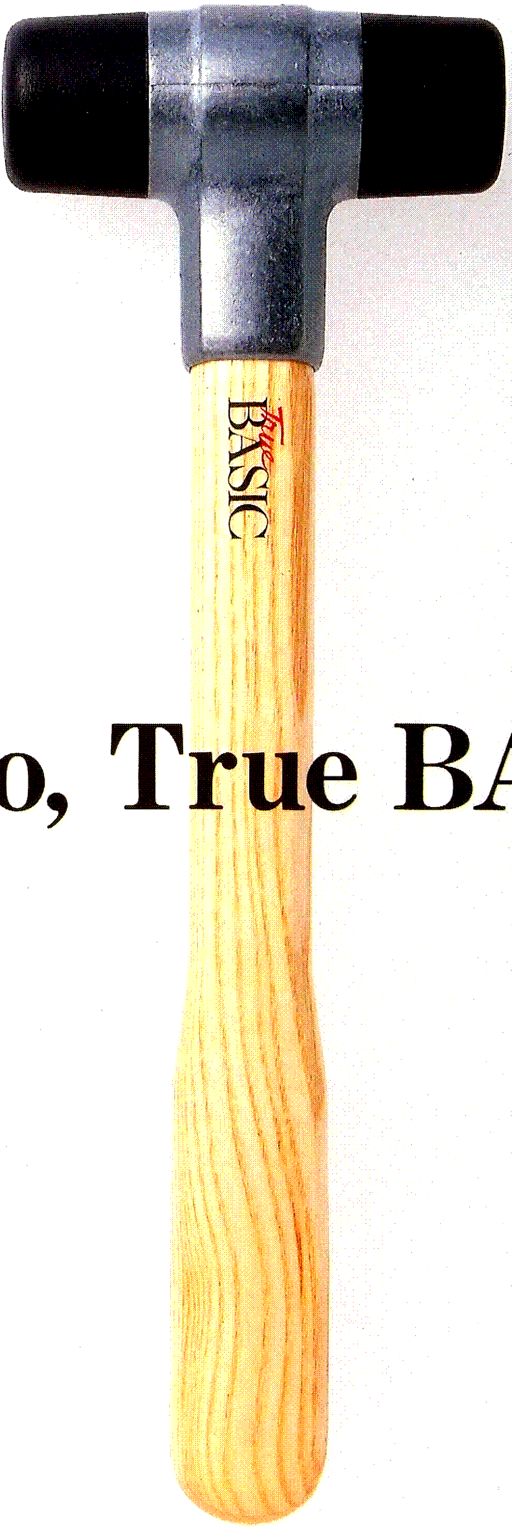
付録:5"2HDフロッピーディスク

本誌掲載プログラム／綴じ込み付録テキストファイル（日本語Microsoft Windows 3.0環境ファイル機能一覧）／クイズ・ランダムスクリーン 解読ツール／バッチファイルコンパイラ「BATCP.COM」／バッチユーティリティ「BU.COM」

SE
Shoeisha Co., Ltd.

創造をクレオらしさと呼びたい

CREO



Hello, True BASIC.

現代の正統派言語　選ぶなら世界標準

1964年、J.G.ケメニーとT.E.カーツの開発した教育用会話型言語「BASIC」は、コンピュータを大変身近なものにしました。すっきりした記述で初めての人でもわかりやすく、慣れればさらに高度に使いこなせる。当時としては、画期的な言語の登場です。こうして生まれ、多くの支持を得たBASICは、ケメニー、カーツ両氏の手によりさらなる進化を遂げました。それが、True BASICです。

①すっきりとわかりやすい、構造化プログラミング

True BASICは、コンピュータが初めての人にもわかりやすい構造化プログラミング。行番号はいっさい使わず、あなたの頭の中のアルゴリズムどおりに、プログラムは組まれます。基礎的な設計力が自然と身につくプログラミング法です。

②初めてでもとまどわない、一貫した操作性

例えば、フルスクリーンエディタとコンパイラとの完全統合。ソースコードの書きこみからデバッグ、コンパイルまで、作業は中断なく滑らかに進行。しかも、作成されたプログラムは直接実行ファイルへ変換でき、手間がかかりません。そのうえ、作業の見とおしがよいマルチウィンドウをサポート。テキストやグラフィックスの作成がスムーズに行なえます。

③シンプルで使いやすい、モジュール(機能)群

機能の高さだけでなく、操作性を重視したモジュール群。作業の流れを中断しない、系統立てた構成からなります。複雑な演算に強い、行列関数など豊富な関数。高速処理に威力を発揮する、数値演算コプロセッサ。よく使う実戦的機能の他にも、仮引数、再帰呼出処理といった高度な機能を満載しました。さらにグラフィックス機能が充実。各種ツールキットや組み込み関数を用いて、簡単作成できます。

④機種もOSも選ばない、優れた互換性

True BASICは、ANSI規格のFULL BASICに準拠。いわば世界標準のBASICです。そのうえ、ハードウェアに依存しない言語構造ですから、異なる機種間、MS-DOSからUNIXまでOS間の互換性があります。つまり、True BASICで構築されたプログラムは、特に変更せずに様々なマシン、OS上で活用できるということ。作成したプログラム資産を無駄にしません。

⑤True BASICは、すでに世界で活用中

対応機種	パソコン版	IBM	PC、PS/2
		Apple	Macintosh 512K、Plus SE、Macintosh II
		Atari	ST 520、1040、MegaST
		Commodore	Amiga 500、1000、2000
	UNIX版	Silicon Graphics Sun Microsystems, Inc. Data General Corp.	
採用している大学	全米で1600校以上		

多彩な機能がいろいろ選べる、ツールキット発売中!

●ビジネス用グラフィックスツールキット
価格：18,000円(税別)

ビジネス用グラフィックスツールキットは、17種類のビジネスチャートを高速描画できる、True BASIC専用のツールキットです。
棒グラフ・線グラフなどの単一グラフから、多種類のグラフを1枚の複合グラフとして表示するなど、多彩な表現でビジネスチャートが作成できます。

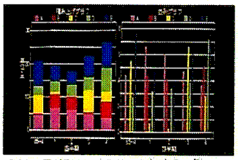
●数学者用ツールキット
価格：25,000円(税別)

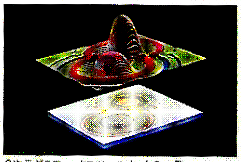
数学者用ツールキットは、10万桁の演算や複素行列など、通常では処理できない複雑な演算を行うことができます。高い精度でデータを求めたい時や、大きな桁数の数値を扱う場合などに、抜群の威力を発揮します。

発売予定のツールキット

- ●3次元グラフィックスツールキット　近日発売
- ●科学技術用グラフィックスツールキット　近日発売
- ●統計用グラフィックスツールキット
- ●文字列操作ツールキット
- ●並べ替え・検索用ツールキット

True BASIC Tool Kit


ビジネス用グラフィックスツールキットの一例


3次元グラフィックスツールキットの一例

新発売

True BASIC™ シンプルだから使える

PC-9800 Series
標準価格 18,000円(税別)

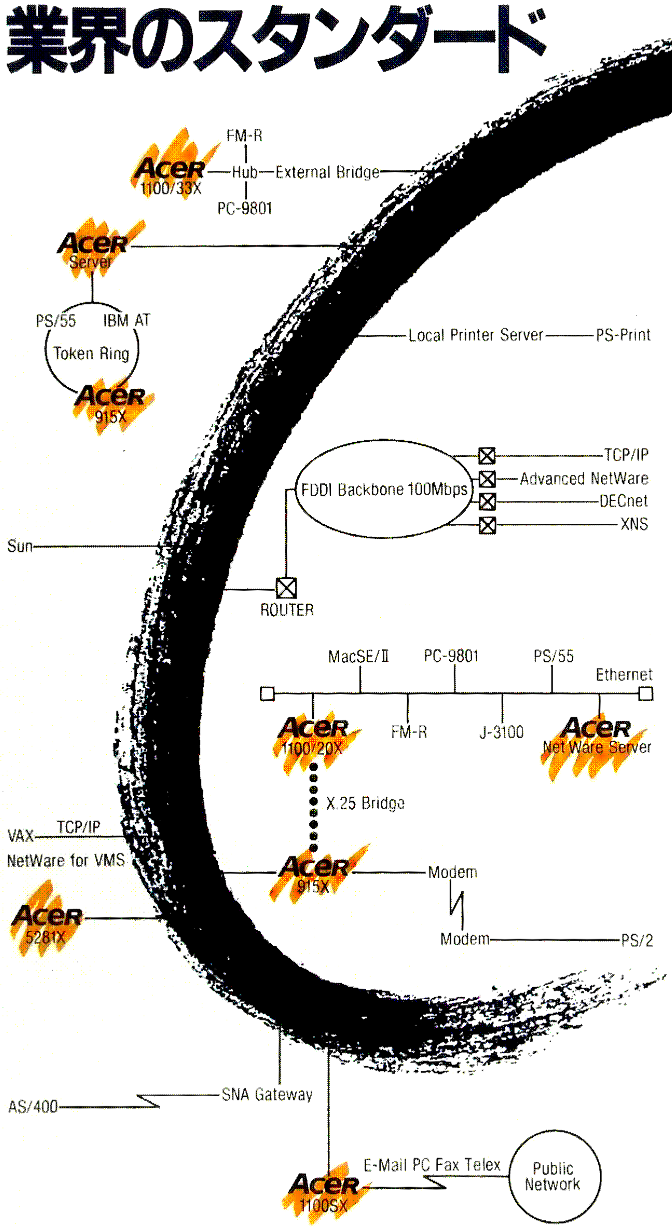
株式会社クレオ
〒108 東京都港区芝浦4-9-25 芝浦スクエアビル TEL03-3456-4650 FAX03-3456-4630

〈資料請求番号P02〉

NetWare

Acer The Word for Value

パソコンLANにおける業界のスタンダード



THE ARCHITECTURE EXTENDED—AX

世界のパソコンLAN市場をリード

米国Novell社の"NetWare"は、世界のパソコンLAN市場において60%以上の圧倒的なシェアをもつLAN OSです。

このNetWareに日本語が付加されたことによってACERなどのIBM-PC(AT互換機)はもちろんのことAXマシン、PC-9801、FM-R、PS-55、J-3100などの国産PCへの対応も広がっており、日本語・英語両環境でのネットワーク構築を可能といたしました。

NetWareの特徴

NetWareは、ミニコンやメインフレームでしか提供できなかったさまざまな機能をパソコンベースで実現することができます。他のLANシステムと比較した場合の大きな特徴の一つにシステム・フォルト・トレラント(SFT)機能があげられます。例えばファイルサーバーに2台のハードディスクを装着させることによって"ディスクの二重化"が可能であり、常に同じ情報が2台のハードディスクに格納されるため、万一メインのハードディスクに故障が生じても、サブ側のハードディスクが動作し続け通常業務に影響を与えることはありません。(ディスク・デュープレックス機能)

また、書き込みのたびに、そのセクターを読み出し、セクターに不良があるとわかったら、以後そこを使わないようにする不良セクター対策(Hot Fix機能)やOSにとって重要なディレクトリー・テーブルとファイル・アロケーション・テーブルについては、それぞれ別の場所に書き込んでおき、読み出しの際には、コピーと比較してエラー検査をします。(ディレクトリー構造の二重化、ファイル・アロケーション・テーブルの二重化など…)

以上のようにセキュリティと障害対策が他社製品に比べて非常に強力であることがNetWareの特徴といえます。

●NetWareとTCP/IPの共存が可能

TCP/IPプロトコル、PC/NFS機能によりUNIXマシンおよびSUNなどのワークステーションをサーバーにするシステムをサポート。これによってNetWareとTCP/IPソフトが同時に共存可能になりました。

●NetWare製品ラインアップ

「NetWare V.2.2英語版」「NetWare V.2.2日本語版」
「NetWare 386 V.3.1J」「NetWare 386 V.3.11英語版」
以上のNetWareシリーズを取扱っております。なお「NetWare 386 V3.1J」に関しましては、接続可能なPCが他のNetWareと異なりますのであらかじめご確認ください。

世界の舞台で磨かれた

Acerファイル・サーバー Acer1100/33 (PC/AT互換機)をベース

- i80386(33MHz)CPU,RAM 4MB
- 5.25インチ(1.2MB)FDD×1基、
- 100MB～680MB HDD最大2基 実装可能
- 14インチモノクロモニター、モニターI/Fカード
- 16ビットバス用 Ethrnetカード1枚 標準装備
- フロアスタンド他

＊Net WareはNOVELL Inc.の登録商標です。

丸文株式会社

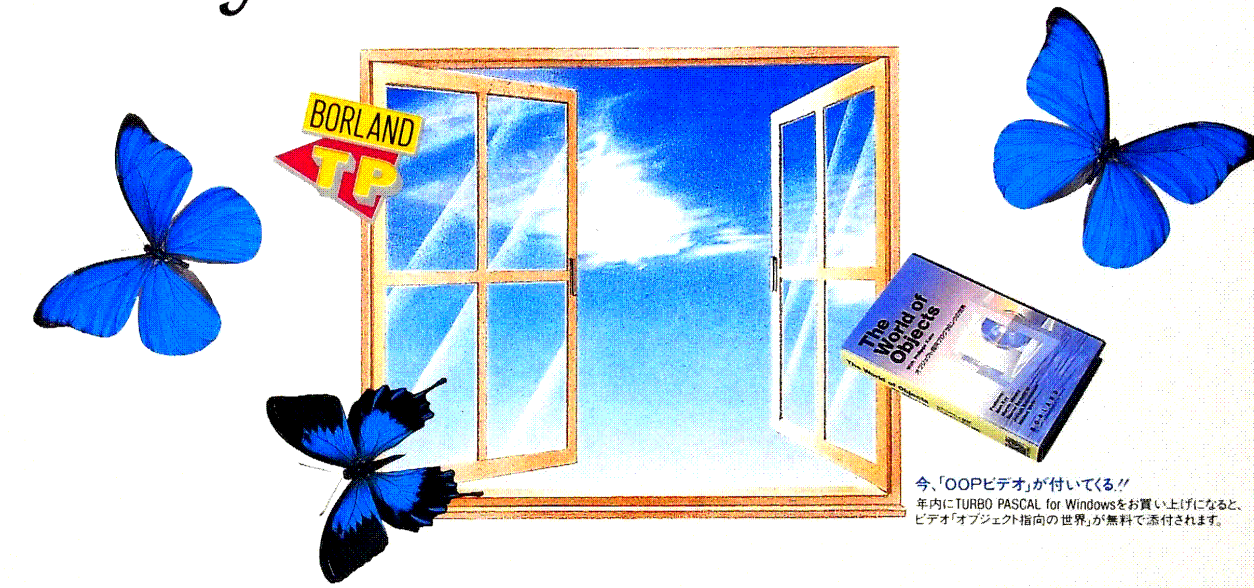
東京都中央区日本橋大伝馬町8-1 丸文ダイヤビル 〒10　第2営業本部 営業第3部 PC&ネットワークグループ TEL (03)3639-9897(ダイヤルイン)

〈資料請求番号001〉

BORLAND

The Leader in Object-Oriented Programming

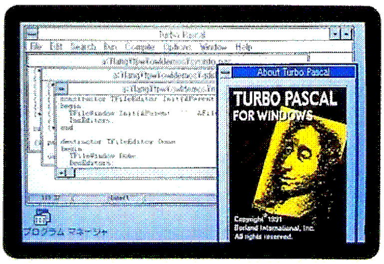
"Easy & True" for Windows.



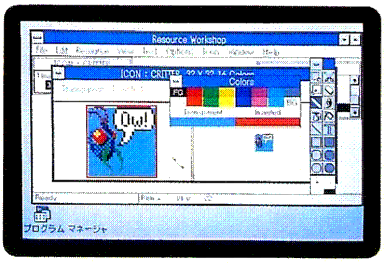
今、「OOPビデオ」が付いてくる!!
年内にTURBO PASCAL for Windowsをお買い上げになると、ビデオ「オブジェクト指向の世界」が無料で添付されます。

基本は、ターボパスカル——。
完ぺきなWindowsアプリケーション開発環境を提供。

その美しい言語仕様から"コンパイラの貴婦人"と呼ばれ、多くのプログラマーを魅了してきたパスカルが今、「Turbo Pascal for Windows」として生まれ変わりました。デビュー当時、その驚異的な低価格とコンパイルスピードの速さ、デバッガやエディタを備えた世界初の統合環境で人々を圧倒し、瞬く間に標準の地位を確立して以来、ターボパスカルは常に先進の機能を搭載し、言語の世界をリードし続けています。アメリカでは、大学などの教育機関や研究施設でターボパスカルを使用することが常識ですし、ヨーロッパや日本でもビジネスの分野で幅広く利用されています。プログラミングの学習もターボパスカルから始めれば、ベーシックで貧弱な言語で学習したために時間を無駄にする、ということはありません。しかもそのターボパスカルが、複雑で難解だったWindowsのプログラミングを簡単にしてくれるのです。



Turbo Pascal for Windowsには、Object Windowsというアプリケーションの骨格になるクラスライブラリが付属しています。ここからウィンドウやメニュー、ダイアログ、コントロールなどを継承すれば、最初からコードを記述する必要はないのです。オブジェクト指向の拡張がされていないC言語でのプログラムより、格段に素早く、短いコードでアプリケーションを作成できるようになります。Turbo Pascal for Windowsは、自分自身がWindowsのアプリケーションであり、Windowsプログラムのみの作成用に設計されています。

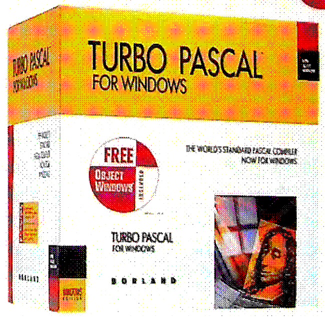


ソースコードの編集、コンパイル、実行、デバッグなど、使いやすさで定評のある統合環境を用い、全てWindows上で行えます。ダイアログ、メニュー、ビットマップ、カーソル、アイコン、文字列リソース、アクセレータ、フォントなどのWindowsリソースを視覚的に作成したり、編集することができるリソースワークショップも付属していますから、高額なSDKを別途購入する必要もありません。Turbo Pascal for Windowsには、Windowsアプリケーション作成に必要なツールの全てが、ひとつのパッケージに含まれているのです。

■動作環境：PC-9800シリーズ（ハイレゾモード対応、LTは除く）、PC-286/386シリーズ/MS-DOS：Ver 3.1以上/MS-Windows 3.0以上必須/マウス：バスマウス/要ハードディスク

■価格……………………29,800円（税別）

NEW



TURBO PASCAL®
FOR WINDOWS

ユーザ向け情報誌
LANGUAGE EXPRESSプレゼント!!

■この広告をご覧になって、製品に関する資料を請求していただいた方全員に、ボーランド発行の情報誌「LANGUAGE EXPRESS」をさし上げます。

※All Borland Products are trademarks or registered trademarks of Borland International, Inc. ©Borland International, Inc.
※Microsoft、Windowsは米国マイクロソフト社の登録商標です。その他、商品名は一般に各社の商標です。
●製品に関するお問い合せは、総販売元(株)マイクロソフトウェアアソシエイツ TEL.03-3486-1411までお願いします。

株式会社ボーランドジャパン
〒107 東京都港区南青山7-8-1 小田急南青山ビル TEL.03-3486-1400(代)

〈資料請求番号002〉

もう640Kバイトの壁に悩まされることはありません。

NDP FORTRAN386は、FORTRAN77のフルセットコンパイラであり、ANSI X-3.9-1987とDOD MIL STD1753、さらにUNIX4.2BSDの拡張機能を完全に満たしているため、ミニコンやワークステーション、メインフレームのプログラムの移植が容易です。また、きわめて高速なオブジェクトを生成します。特に配列変数のアクセスはDOS上のコンパイラの数倍から10数倍のスピードのオブジェクトを出力します。インラインコードの生成も可能です。

- ●実装メモリ分だけ配列を宣言することができます。
- ●シンボル名として31文字まで使用できます。$の使用も可能です。
- ●INCLUDEのネストが可能です。
- ●MIL STD-1753のビット操作をすべてサポートしています。
IOR、IAND、IEOR、NOT、ISHFT、ISHFTC、ISHFTL、ISHFTR、BTEST、IBSET、IBCLR、MVBITS
- ●ベクセル関数、ガンマ関数、誤差関数をサポートします。
- ●ZとO編集が入出力に使用できます。
- ●FORTRAN66のDOループをオプションスイッチにより選択できます。
- ●カンマを使用したフリーフォーマット入力をサポートしています。
- ●$を入出力の改行文字として利用できます。
- ●カラム1にX, Dを記入することにより、条件付きコンパイルができます。
- ●COMMONとEQUIVALENCEにおいて文字と数字を混在できます。
- ●コマンドラインに指定したGETARG、IARGC、GERENVを読み込む機能をサポートしています。
- ●次の型が使用できます。
REAL＊4、REAL＊8、INTEGER＊1、INTEGER＊2、INTEGER＊4、LOGICAL＊1、LOGICAL＊2、LOGICAL＊4、COMPLEX＊8、COMPLEX＊16
※デフォルトのLOGICAL、INTEGERは4ですが、それを2に変更することもできます。

定価　¥148,000

80386
プロテクトモード対応

NDP FORTRAN 386

32ビット専用
フォートラン
コンパイラ

動作環境

必要なハードウェア

■パソコン本体

NEC
- ●PC-9801シリーズ（XL2、RA、LS、ES、RA21/51、RS21/51、T、DA、DS、NS）
- ●PC-H98シリーズ（Model 60、70、100）

EPSON
- ●PC-386シリーズ（386、LS、V、M、G）

東芝
- ●J-3100（SGT、SGX、GXS）、J-3300

IBM
- ●PC/AT互換機
- ●各種AXマシン（80386、80386SX、80486搭載機）
- ●ハードディスク（5MByte以上の空領域）
- ●2MByte以上のプロテクトモードメモリ
- ●数値演算コプロセッサ（80387/387SX、またはWeitek3167/TURBO-3167）

必要なソフトウェア

■MS-DOS

NEC、EPSON MS-DOS ver3.3以上
IBM PC/AT機　PC-DOS ver3.2以上
東芝J-3100等　日本語DOS ver3.1
英語DOS ver3.3

80386CPUのプロテクトモード専用プログラム開発キット

■386|TOOLBOX

386|DOS-Extenderのほか、32ビット対応のアセンブラ、リンカ、ライブラリアン、デバッガを一つにまとめたパッケージで、386/486の全ての命令セットを使用したアプリケーションの開発を可能にします。ユーザメモリは640Kバイトだけでなく、実装されている全てのメモリを利用することができます。MS-DOSの環境を利用しているので、従来のエディタなどをそのまま利用できます。なお開発した製品を販売する際はランタイムライセンス（¥600,000）が必要となりますが、開発のみの場合にはまったく必要ありません。

定価　¥148,000

32ビットプロテクトモード専用

開発ツールキット

386|TOOLBOX ＋ 386|DEBUG ＋ 386|VMM	￥248,200
386|TOOLBOX ＋ 386|DEBUG	￥175,100
386|TOOLBOX ＋ 386|VMM	￥198,900
386|DEBUG ＋ 386|VMM	￥129,600

テックソフト&サービス株式会社
コンピュータ事業部
〒273 千葉県船橋市本町7-11-5 レランドセンタービル5F
TEL.(0474)23-1322 FAX.(0474)23-0202
ユーザサポート専用電話 TEL.(0474)22-9355 担当：菅原

※すべての価格に消費税は含まれていません。

BORLAND
The Leader in Object-Oriented Programming

問：Windows™アプリケーション開発に最も有効なコンパイラは、何ですか？

（解答はページ下をご覧下さい）

(Inactive C:¥WORK¥BORLAND.EXE)

	従来の開発環境	？
	¥ 98,000	¥ 98,000
Cコンパイラ		含む
C++コンパイラ	なし	含む
Windows AP作成ツール	¥ 98,000	含む
アセンブラ	¥ 40,000	
合計	¥236,000	¥ 98,000

※上記の価格には、消費税は含まれていません。

パソコン上に、視覚的で快適な操作環境を提供するWindows 3.0。そこで利用するアプリケーション作成のためのコンパイラは、現在のものでベストといえるでしょうか？機能、スピード、価格…。これらの面で理想的ともいえるコンパイラが、登場しました。「BORLAND C++」です。BORLAND C++は、Windowsアプリケーションを開発、デバックするC/C++コンパイラとツールを標準で装備しています。

■Windowsアプリケーション開発をサポート

BORLAND C++は、Windowsプログラムを完全にサポート。高速かつ高性能な処理を実現しています。MDI(Multiple Document Interface)やDLLs(Dynamic Link Libraries)といったようなWindowsの機能を使って、大規模なプロテクトモードアプリケーションの開発が可能になります。さらに、MicrosoftのリソースコンパイラとWINDOWS.H、Resource Work Shop(日本語版)が標準装備されていますので、別途の開発キット(SDK)は不要です。

■ANSI C／AT&T C++両用コンパイラ

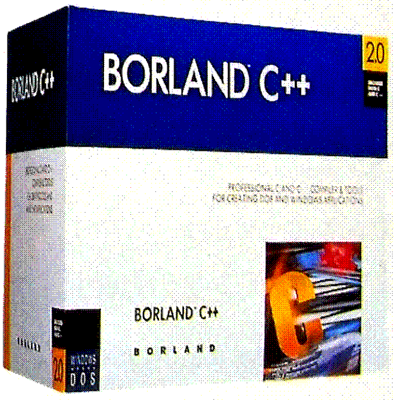
BORLAND C++は、CとC++両方のコンパイラとして利用できます。さらに、再コンパイルの速度を大幅に向上させる「コンパイル済みヘッダ」機能をサポート。MS-DOSの640KBというメモリの制限を破るオーバーレイマネージャVROOMの利用や、Turbo Driveによってプロテクトモードで動作する統合開発環境での大規模なプログラムの作成も可能です。

■Programmer's Platform

Programmer's Platformは、好きなツールを組み込む機能を持つ強力な統合開発環境で、MS-DOSやWindowsアプリケーション開発に必要なすべてのツールがすでに組み込まれています。さらにオーバーラップウインドウやマウス、そしてWindowsのプログラミングリファレンスを含んだオンラインヘルプをサポート。内蔵のマルチファイルエディタは、アンドゥ(実行の取消)／リドゥ(再実行)やマクロ機能を装えています。

■動作環境：PC-9801シリーズ(ハイレゾモード対応、LTは除く)、PC-286/386シリーズ／メモリ：640KB以上／MS-DOS：Ver 3.0以上／MS-Windows 3.0以上／マウス：バスマウス／要ハードディスク

■価格……………………………**¥98,000**(税別)



ユーザ向け情報誌
LANGUAGE EXPRESSプレゼント!!
■この広告をご覧になって、製品に関する資料を請求していただいた方全員に、ボーランド発行の情報誌「LANGUAGE EXPRESS」をさし上げます

（答）BORLAND® C++

※All Borland Products are trademarks or registered trademarks of Borland International, Inc. ©Borland International, Inc
※Microsoft、Windowsは米国マイクロソフト社の登録商標です。その他、商品名は一般に各社の商標です
●製品に関するお問い合せは、総販売元㈱マイクロソフトウェアアソシエイツ TEL.03-3486-1411までお願いします

株式会社ボーランドジャパン
〒107 東京都港区南青山7-8-1 小田急南青山ビル TEL.03-3486-1400㈹

〈資料請求番号004〉

Brief

Full Screen Editor 日本語版

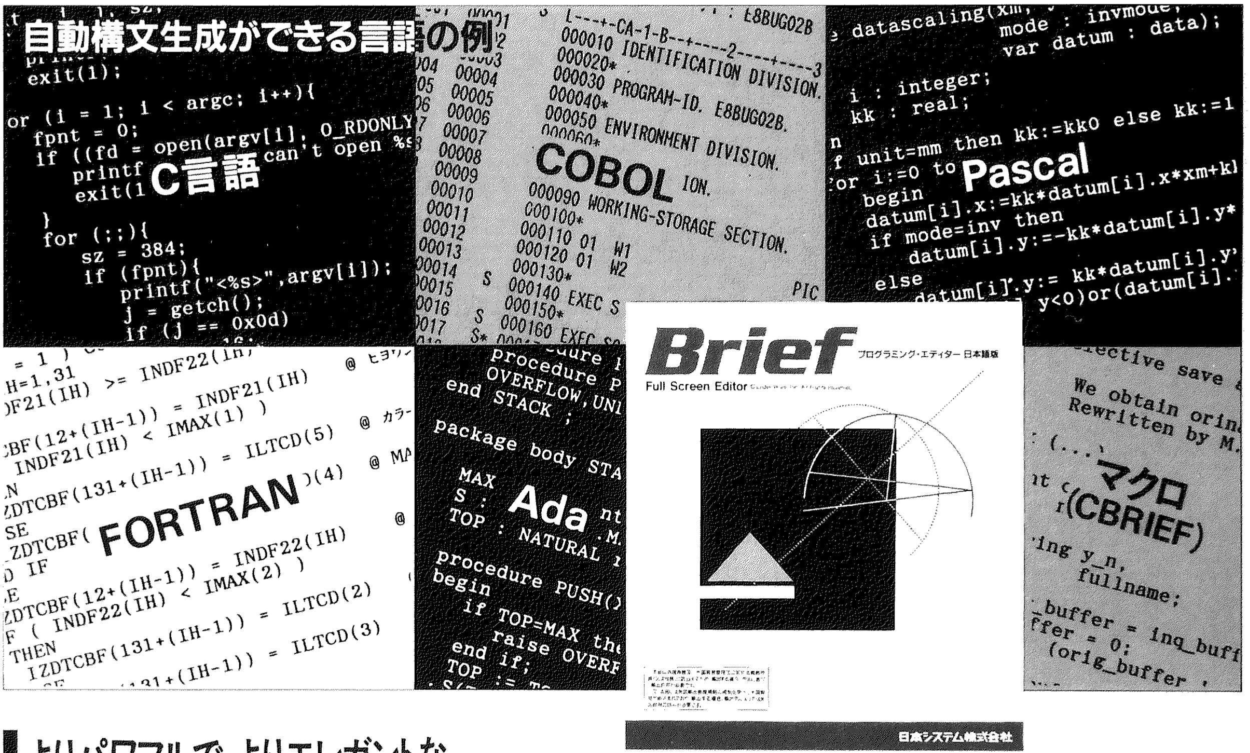
全米のプログラマが認めた実力！
アメリカでは「ブリーフ」が
プログラミング用エディターとして
最も多く利用されています。

C言語で組めるマクロ！

V.3.0 いよいよ登場。

Version UP!

プログラミングエディター新世紀。



よりパワフルで、よりエレガントな プログラミングのために……。

●C言語で組めるマクロ 新機能

使い慣れているC言語でマクロが組めますから、習得が簡単でカスタマイズも思いのままです。

●マクロのスクリーン・デバッガ 新機能

ソースコードでデバックできます。もちろんブレークポイントやウォッチ・ドッグもセットできます。

●テンプレート機能〈構文の自動生成〉 新機能

C、Ada、Basic、COBOL、FORTRAN、MODULA2、Pascalで構文を自動生成します。

●300回ものアンドゥ

各ファイルごとに300までの操作を元にもどすことができますので、やり直しの面倒を気にせず、安心して編集できます。

●安全編集

Briefは、一定時間キーを操作しないこと最新の編集内容をファイルに自動保存します。だから編集中に電源が切れても安心です。

●その他の主な新機能

キー操作のファイルへのセーブとリプレイ、処理可能行数を拡張、リジューム機能、Ctrlキー・アサイン（オプション）、メニュー（オプション）etc.

アメリカでも日本でも高い評価

- ●プログラミング向きエディターを探している人は、もう探す必要はない。（Jerry Pournelle BYTE）
- ●ソフト開発には強力なツール、300回のアンドゥは大きな特徴。（日経バイト）
- ●既存のエディターに不満を覚えたら、これしかない。（Oh! FM）
- ●まるで夢のようなアンドゥだ――おもわず間違うのを期待してしまいそうだ。（The C Users Journal）

■主な機能

◆55ウィンドウ◆無限のファイル◆自動コンパイル◆自動エラー・ジャンプ◆自動インデント◆自動テンプレート◆バッファ書出 切換 削除 リスト表示◆ファイル名の指定 表示 変更◆バックアップ保存◆特殊文字入力◆2行を1行に連結◆ファイル単位の挿入◆増分検索◆各種マーク 始・終点の入換◆ブロックのカット コピー ペースト 印刷 保存 左右へのシフト◆対応する括弧へ移動◆C言語のコメント化 非コメント化◆DOSコマンドの実行◆キー操作記憶 実行◆入力ヒストリー◆英文テキスト整形◆タブのスペース挿入◆拡張子毎に異なった設定ができるタブ◆検索置換パターンの画面からの入力◆英字の大小文字変換 その他

定価48,000円
（消費税は含まず）

■5機種対応パッケージ

（東芝、NEC、AX協議会、IBM、富士通）

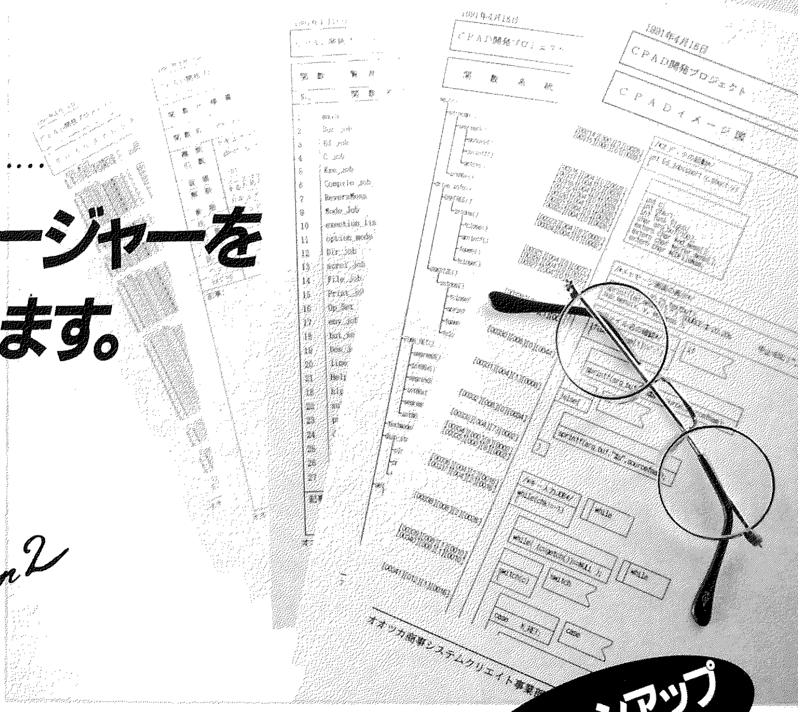
●5.2DD、3.5 2DD デュアルメディア●MS-DOS 3.1以降に対応●英語DOS では使用できませんが、英語版 BRIEFもセット販売いたしております。●お急ぎの方、上記以外のメディア希望の方、英語BRIEF のセット販売希望の方はユーザー・サポート係までご連絡下さい。

日本システム株式会社

●お問合せは 東京都立川市曙町1-18-2一清ビル立川事業所Briefユーザー・サポート係
TEL0425-23-4469 FAX0425-27-6127

〈資料請求番号005〉

COMPUTER PRO-STAFF
SYSTEM CREATE



すべてのCプログラマのために……

プロジェクト・マネージャーを強力にサポートします。

C言語開発支援ツール

CPAD Version 2

シーパッド Ver.2

98シリーズ対応版……¥85,000（税抜）
ダブルメディア 3.5／5インチ添付

高品質のドキュメンテーションを驚くほど簡単に実現する、新世代のプロジェクト指向グループウェア「CPAD V.2」登場。

バージョンアップ 機能強化

CPAD Ver1.5ご購入登録ユーザ様は 無償バージョンアップ（送料のみご負担いただきます。）

「CPAD」はこれまでC言語による開発でソフトハウスが悩んでいた開発時の工数短縮及び複数人数によるプロジェクト管理を容易にいたします。また、既存のソースリストの仕様書を自動作成しますので、管理体制を容易に画一化することが可能となり、プログラマのドキュメンテーションの負担を軽減することができます。

6大機能Cソースの解析、援助、学習ツールとして

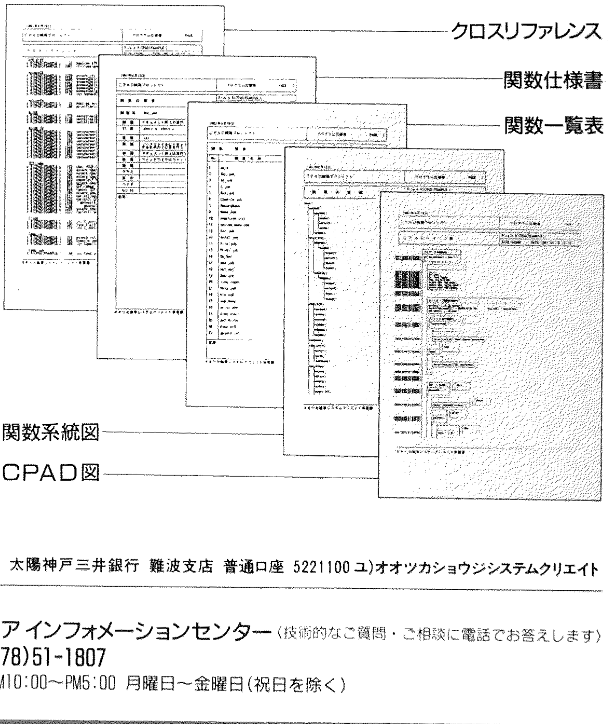
①C言語ソースのドキュメント管理を自動化、一律化
②全ドキュメントはソース上で統括管理
③CPAD統合環境モード新規登載
④カーニハン&リッチーの書式にソースをリフォーム
⑤PAD様式に準ずる出力機能
⑥エディタ編集可能なドキュメントと印刷スタイル設定機能

動作環境

①対応パソコン	NEC PC9801シリーズ、EPSON PC-386・PC-286シリーズ
②対応OS	MS-DOS V2.1以上
③メモリ	640Kバイト以上
④周辺機器	NEC 201サポートのプリンタ。 ハードディスク、RAMディスクなど MS-DOS上で利用可能なメディアであれば、何れも可
⑤ディスプレイ	640×400であれば、何れも可。
⑥対応C言語	ANSI準拠によるC言語コンパイラ用ソースであれば、何れも可。

● MS-DOSは米国マイクロソフト社の商標です。（本製品にはMS-DOSは含まれておりません）

自動出力印刷例



※お求めの際は有名パソコンショップ、または右記口座にお振り込みのうえ、電話にてご注文下さい。太陽神戸三井銀行 難波支店 普通口座 5221100 ユ)オオツカショウジシステムクリエイト

オオツカ商事 COMPUTER PRO-STAFF SYSTEM CREATE
本社：〒760 高松市西内町5-14オオツカビル内

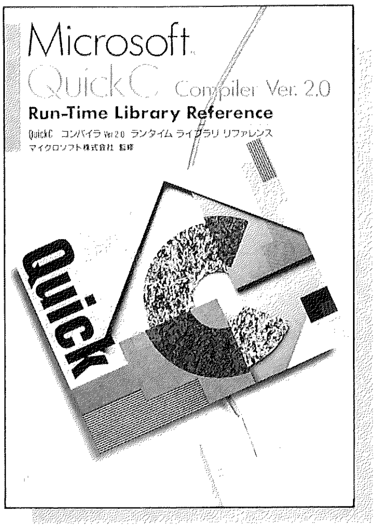
ソフトウェア インフォメーションセンター（技術的なご質問・ご相談に電話でお答えします）
TEL.(0878)51-1807
受付時間 AM10:00～PM5:00 月曜日～金曜日（祝日を除く）

外部スタッフ、及び正社員募集中!! 近距離、遠距離を問わず 詳しくは人事部直通電話(0878)51-1785まで

〈資料請求番号006〉

プログラミングのための優良図書

SE
Shoeisha Co., Ltd.

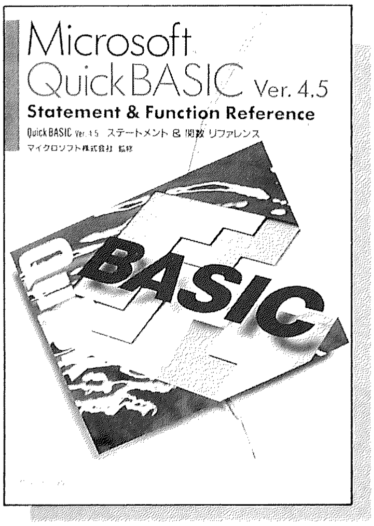


Quick C コンパイラ Ver.2.0 ランタイムライブラリリファレンス

Quick C Compiler Ver.2.0
Run-Time Library Reference

マイクロソフト株式会社　監修
A5判/364頁
定価2,400円

『言語ソフトのベストセラー「Microsoft Quick C」のオンラインヘルプを完全書籍化。マイクロソフト社監修によるパーフェクトQuick C リファレンス。プログラミング時の参考書として、また学習書として活用できます。』



Quick BASIC Ver.4.5 ステートメント&関数リファレンス

Quick BASIC Ver.4.5
Statement & Function Reference

マイクロソフト株式会社　監修
A5判/394頁
定価2,400円

『Quick BASIC Ver.4.5のQBアドバイザを完全書籍化。パソコンの前を離れて参照したい時など学習用として最適です。各ステートメント & 関数にサンプルプログラミングを満載したパーフェクトリファレンス。マイクロソフト社監修。』

Quick BASIC スタイルブック

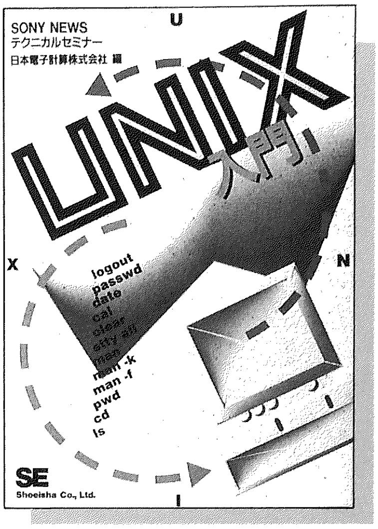
イント・ツーワン　著
B5判変型/236頁
定価2,300円

『初心者からマニアまで幅広く使われるQuick BASIC。このQuick BASICを用いて質の高いプログラミングを書くためのノウハウがいっぱいつまっています。一問一答の問題集形式。パズルのようにプログラミングが学べます。』

UNIX入門
―SONY NEWSテクニカルセミナー―

日本電子計算株式会社　編
A5判/237頁
定価1,800円

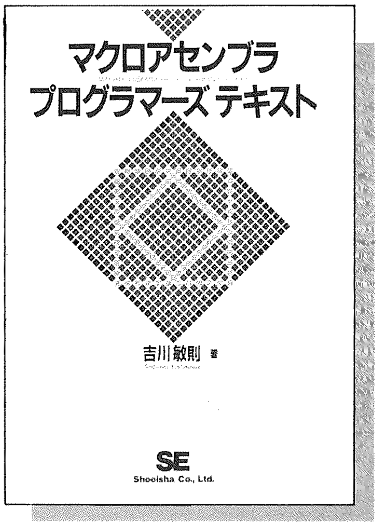
『ソニーのNEWSによる「UNIX入門」セミナーのテキストをベースとした入門書。UNIXのコマンド、エディタの使い方、日本語入力の方法、ネットワークの利用方法などを豊富な実例をまじえて解説。はじめてのワークステーションを利用するユーザ必携の書。』



マクロアセンブラ プログラマーズテキスト

吉川敏則　著
A5判/243頁
定価2,600円

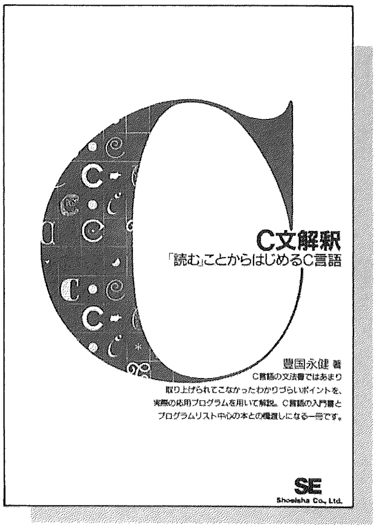
『最低限必要なハードとソフトの知識から、データ演算の仕組みを踏まえた上で、アセンブラプログラミングの基本を丁寧に解説。アセンブリ言語に興味を持つ人や、これからアセンブリ言語でプログラムを作成する人に役立つ一冊。』



C文解釈
―「読む」ことからはじめるC言語―

豊国永健　著
B5判/183頁
定価2,000円

『C言語の文法書ではあまり取り上げられてこなかったわかりづらいポイントを、実際の応用プログラムを用いて解説。C言語の入門書とプログラムリスト中心の本との橋渡しになります。』



SE Shoeisha Co., Ltd. 株式会社翔泳社 〒102東京都千代田区平河町2-4-14平河町KSビル TEL.03-3263-0447(代表)
翔泳社の本は全国どの書店でもお求めになれます。店頭にない場合は「至急」とお申し出の上ご注文いただければ10日位でお届けできます。

Programmer's Page

MONTHLY MAGAZINE FOR PROGRAMMERS

特集

Windowsプログラムの作り方

VisualBASIC, TPW, BC++など新処理系登場!

SDK+MS-C の標準的開発環境を理解するために
Windows API はこうして使う　永井知己　2

Windows でも威力を発揮するオブジェクト指向言語
Borland C++　大野元久　21

クラスライブラリを使ったオブジェクト指向プログラミング
Turbo Pascal for Windows　有澤雄志　41

ビジュアルなプログラミング環境を実現する
VisualBASIC　海老原浩之　51

ウィンドウ環境を生み出したGUIの元祖
Smalltalk/V Windows 1.1　60

クラスライブラリ, リソース作成, SQL処理など
Windows プログラム開発支援ツール　64

プログラム「遊」工房　十乃巻
「SNAKE」ゲーム　谷口典生　68

IC 千夜一夜　第5回
メモリの IC 化　―D-RAM LSI の発展―　虎岩登　78

MS-DOS ツール・ラボ　第8回
リングバッファの利用・起動記憶プログラム　TIMERING.EXE　加野島英渡　82

実践C塾　基礎テクニック篇　第5回
プリンタ出力　大野元久　92

98 活用アセンブラ技法　タイマ編　第1回
タイマ BIOS の補強　星野操　102

付録

5" 2HD フロッピーディスク

綴じ込み付録

日本語 Microsoft Windows 3.0環境ファイル機能一覧

表紙イラスト：日暮修一
表紙 AD：佐々木幹夫(SE)

パソコンプログラマのためのクリエイティブマガジン

月刊プログラマーズ・ページ

パソコン博物館　第11回
新技術の開拓者／富士通 FM8　井戸禎光　74
プログラマの道具学　第11回
コーヒー　祐安重夫　76
PRODUCTS REVIEW
日本語 TrueBASIC V3.0J　113
情報処理技術者試験　第２種合格ノトレーニング
第８回　COBOL　白井儀巳・佐野有一　117
Book Review「プログラマのうちあけ話」　中村正三郎　122
本のアンテナ　古田島義和
佐々木哲のニューヨークシステム通信　124
西川和久の希望的観測　126
コンピュータの法律　熊谷信太郎　128
アキバ短信
プログラマの仕事場　第６回　マイクロプローズジャパン株式会社　130

INFORMATION PAGE　133
NEWS & NEW PRODUCTS　134
Programmer's Program　ZZ386　138
オンラインソフト採集帳11「BATCP」　140
HEALTHY? UNHEALTHY?
ドラゴン・ポスト　142
付録ディスクのご案内　144
バーチャルスペースガイド　145
読者プレゼント
クイズ・ランダムスクリーン　146
オンラインサービス（SE-NET）のお知らせ
バックナンバーのご案内　147

広告索引　資料請求番号

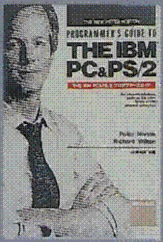
人材募集広告

(株)エルテックス
(掲載順)

1991 NOVEMBER No.11

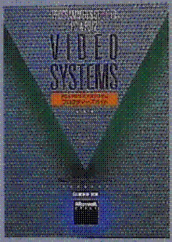
翔泳社が贈るプログラマ必携書

IBM PC & PS/2 プログラミング 3部作



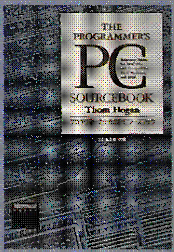
あのピーター・ノートンのベストセラー。IBM プログラマ必携のスタンダードガイダンス。

IBM PC & PS/2 プログラマーズガイド
● Peter Norton & Richard Wilton著
●定価　5,200円



IBM PC ファミリーのビデオシステム完全ガイド。

PC & PS/2 ビデオシステム プログラマーズガイド
● Richard Wilton著
●定価　6,000円



IBM PC & PS/2 プログラマ必携の技術資料完訳。

プログラマーのための PC ソースブック
● Thom Hogan著
●定価　5,200円

IBM Authorized Dealer

今月の特選お買い得品！

T·ZONE

80486/33MHz made in USA
先着20台限り **超特価¥498,000**

80486/33MHz
RAM 4MB(max 32MB)
64KBキャッシュ(max 256KB)
3.5"1.44MB×1/5"1.2MB×1
CONNER 120MB HDD
スーパーVGAアダプタ512KB
2シリアル、1パラレル
拡張スロット×8
ドライブベイ×6
ミニタワー
アルプス製101キーボード

80386/33MHz ノート made in TAIWAN
限定20台 **超特価¥498,000**

80386/33MHz
RAM 2MB(max 16MB)
32KBキャッシュ
3.5"1.44MB×1
40MB HDD
32階調VGAバックライト液晶
外部VGA出力
1シリアル、1パラレル
外部テンキー付属
外部101キーボード、外部FDDポート
ニッカドバッテリー(充電2時間、2時間使用)
280W×220D×54H 4kg

Mitac

3060D	101キーセット	106キーセット
RAM 3MB、HDD 40MB	¥318,000	¥333,000
RAM 3MB、HDD 120MB	¥368,000	¥383,000
RAM 5MB、HDD 120MB	¥385,000	¥400,000

共通仕様
80386SX/16MHz、3.5"1.44MB×1、5"1.2MB×1、スーパーVGA(512KB)、スーパーVGAモニター

MIMI-SCSI 超特価¥29,800
プリンタポートにつなぐだけでSCSIデバイスが使えるようになります。
87MB HDDセット 超特価¥138,000

マウス シリアル 超特価¥6,800〜 PS/2 超特価¥8,000〜

PS/55 NOTE用 2MBメモリー 超特価¥28,000〜
4MB 超特価¥48,000〜

Microsoft BALLPOINT MOUSE 超特価¥28,000

VGA信号をビデオに変換するコンバーター 超特価¥98,000
320×200〜640×480の信号をS-VHS、ビデオに変換して、テレビやビデオに接続できます。

SCSI I/F 超特価¥9,800〜 SCSI I/F, W/FDC 超特価¥15,000〜

WINDOWS 3.0
時代のトレンドWINDOWS 3.0がIBMではマルチウィンドウズで複数のDOSアプリが同時実行可能です。

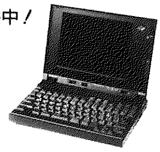
IBM	日本語WINDOWS 3.0	¥20,000
ADOBE TYPE MANAGER		¥20,000
AFTER DARK		¥9,800
BECKERTOOLS		¥16,000
MICROSOFT ENTERTAINMENT PACK		¥8,800

その他WINDOWS用アプリケーションが200種類以上あります。
WINDOWS 3.0対応のPDS(無料)もあります。

NOTE
ノートといえば、PS/55 note 大特価提供中！

IBM	PS/5523SJ1	~~¥248,000~~
IBM	PS/5523SJ4	~~¥368,000~~
NBCC	386NOTE	¥298,000
NBCC	386LAPTOP	¥266,000
NBCC	286NOTE	¥244,000
NBCC	286LAPTOP	¥220,000

PS/55 note



DOS/V
DOS/VといえばT·ZONEです。入門機から互換機まで！

IBM	PS/5530-SW4	大特価提供中！ ~~¥609,800~~
	標準で6MBのRAMと日本語WINDOWS 3.0が付属しています。勿論DOS/Vも付いています。	
IBM	PS/5510ZJ2	大特価提供中！ ~~¥214,000~~
IBM	PS/5510ZJ4	大特価提供中！ ~~¥308,000~~
IBM	PS/5510TJ4	大特価提供中！ ~~¥554,000~~
ALR	POWER FLEX 20CSX, RAM 5MB, 3.5" 85", 40MBHD	¥385,000
ALR	BUSINESS VEISA 486/33, RAM, 2MB, 3.5" 85", 120MBHD	~~¥1,250,000~~
東芝	DynabookでDOS/Vが動いています(2Fでデモ中)	

VGA
DOS/Vの必須アイテムはVGAアダプターです。
豊富な在庫から選べます。

BASIC VGA 640×480	¥19,800
SUPER-VGA 1024×768 16色	¥38,000
SUPER-VGA 1024×768 256色	¥44,800
Dyanbook用VGAアダプタ	¥49,800

VGAモニター
DOS/V対応やWINDOWS用として高解像大画面のモニターがほしいあなた、何を選びますか？

VORTEC CM-1413 14" 0.28mm 15.0-38kHz 1024×768 ¥68,000
デジタル・アナログ両用

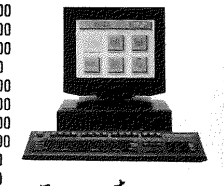
三菱	XC-98V3	14" 0.28mm 98/VGA	640×460	¥78,000
ACER	7015	14" 0.31mm 15.0-36kHz	800×600	¥85,000
日立	CM1483MU	14" 0.28mm 30-40kHz	1024×768	¥89,000
NEC	MULTISYNC II	14" 0.31mm 15.5-35kHz	800×560	¥95,000
NEC	MULTISYNC 3D	14" 0.28mm 15.7-38kHz	1024×768	¥180,000
NEC	MULTISYNC 4D	16" 0.28mm 30.0-57kHz	1024×768	¥280,000
IBM	8515	14" 0.28mm VGA, 8514/A, XGA		~~¥168,000~~

GAME
時代はゲーム!? TERADRIVEや新作のソフトが山盛りでご来店をお待ちいたしております。

SEGA TERADRIVE MODEL1	~~¥148,000~~ 大特価提供中！
SEGA TERADRIVE MODEL2	~~¥188,000~~ 大特価提供中！
SEGA TERADRIVE MODEL3	~~¥248,000~~ 大特価提供中！
SEGA TERADRIVE 専用モニター	~~¥89,000~~ 大特価提供中！
SEGA TERADRIVE マウス	~~¥6,800~~ 大特価提供中！

WALT DISNEYの永遠のアイドル、ミッキーマウスがあなたにしゃべりかけます。

MICKEY'S ABC	¥10,800
MICKEY'S 123	¥10,800
MICKEY'S ABC & SOUND SOURCE	¥13,800
SOUND SOURCE(驚異のローコスト音源スピーカーです)	¥6,800
ADLIB FOR ATバス	¥30,000
ADLIB FOR MCAバス	¥40,000
SOUND BLASTER FOR ATバス	¥48,000
SOUND BLASTER FOR MCAバス	¥68,000
JOYSTICK	¥4,800
ゲームカード(PC/AT)	¥9,800
ゲームカード(MCA)	¥15,000



仕事も 遊びも ちょっと、こだわる。そこで…

世界のソフトウェアが走る。

DOS/V OADG

PS PLAZA
のサポート拠点です！

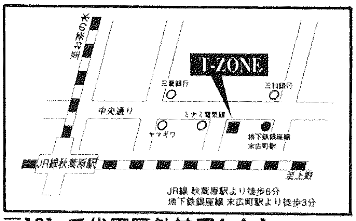
T·ZONEは今年1月19日からDOS/Vのサポート拠点として「PS PLAZA」をオープンしております。

特徴その1 日本IBMのSEが親切にどんなご相談にもお答えします。
特徴その2 いろいろな互換機上でDOS/Vが動くマルチプラットフォーム環境が揃っています。
特徴その3 PS PLAZAの直通番号があります。(03)3257-2654
特徴その4 外人さん大歓迎／英語、韓国語、ポルトガル語、イタリー語、スペイン語、バングラ語などのマルチリンガルOKです。
特徴その5 AT環境のボード、ソフトが日本一／揃っています。
特徴その6 DOS/V、WINDOWS、LAN、マルチメディアに関する、ホットな最新情報が得られます。

お知らせ

DOS/Vプログラムコンテスト開催。
DOS V用のユーティリティーやFEP、表計算ソフト、エディターなど、どんなジャンルのソフトでもOKです。
PDS、フリーウェア、シェアウェアをどしどし作って下さい。
DOS V用であれば何でもOK！
特選1名様に「テラドライブ」をプレゼント……他、お申し込みの方全員にもれなく商品をさしあげます。ふるってご応募下さい。

T·ZONE
IBMゾーン
☎03-3257-2652



3F ハードウェア 〒101 千代田区外神田4-4-1
4F ソフトウェア ☎03-3257-2652 FAX 03-3257-2661

T·ZONEスタッフ募集 IBM、Apple、無線のスタッフを募集中です。アルバイトも募集中！ ☎03-3257-2690

☆T·ZONEの7Fは専門書籍(国内・外)・データーブックの専門フロアです。大量在庫中／ ほしい本は何でも揃います。

〈資料請求番号007〉

特集

Windows プログラムの作り方

VisualBASIC, TPW, BC++など新処理系登場!!

日本語Windows 3.0の出荷本数は，すでに13万本を越えたといわれている．しかし，Windowsアプリケーションは，まだ数えるほどしか市場に登場していない．Windowsアプリケーションの開発が遅れている理由として，プログラマがまだWindowsプログラムになれていない，Windowsのプログラミング環境が整備されていない，などがあげられる．またアマチュアにとっては，投資金額の高さが大きな障害となっていた．しかし，徐々にではあるが，Windows対応の言語処理系も数を増し，Windowsプログラミングの困難さを解決するためのツール類も揃いつつあるようだ．今回の特集では，Windows APIの上手な使用法を解説するとともに，Borland C++，Turbo Pascal for Windows，VisualBASICなど，新たに登場した言語処理系の紹介を行った．

PROGRAMMING

Windows

特集 Windows
プログラムの作り方

SDK＋MS-Cの標準的開発環境を理解するために
Windows APIはこうして使う

株式会社 ANK＆
永井知己

さまざまな処理系が，Windows 対応を売りものにしている感のある昨今だが，その中でも以前からの標準である「MS-Windows ソフトウェア開発キット（SDK）＋MS-C」は，アプリケーション開発環境の本命である．とはいえ，メッセージ駆動型の構造を理解し，数百種類の Windows API を使いこなすのは，なかなか容易なことではない．SDK を使ったプログラミングは「わかりにくい」「複雑な」ものだと，否定的なイメージをもつ読者も少なくないだろう．本稿では，初心者がよく使う，あるいは使うべき Windows API を中心にその機能と使い方を解説するとともに，Windows API の体系的な整理を試みる．また，Windows プログラミングのポイントのひとつであるメニューとダイアログボックスの作成方法，および，SDK 付属ツールであるダイアログエディタの効果的な使い方についても解説する．

注：SKD＋MS-C を用いた Windows プログラミングの基本については，本誌1991年２月号「特集：Windows の時代である！」で紹介しています．なお，1991年２月号のバックナンバーが品切れとなっているため，２月号をおもちでない読者のために，本稿と関連する「Windows 基本プログラム大解剖・永井知己」を付録ディスクに収録しています．

Windows API関数使いこなしのコツ

これだけは覚えたい基本API

Windowsには，約600ほどのWindows API関数が用意されています．はじめて，Windowsアプリケーションを作成するときは，APIの数だけで圧倒されてしまうかもしれませんが，実際にWindowsアプリケーションを作成するときに，必ずしも全部が必要というわけではありません．すべてのWindows APIを暗記しようなどと思えば，挫折するのは当然ともいえます．これを，内容からいくつかのグループに分類し，まとまった体系として整理すれば，はるかに理解しやすくなります．

また，Windowsに関するちょっとした知識があるだけで，覚えなければならない関数の数はぐっと少なくなります．私の経験からいって，WindowsのAPIは大きく次の5種類に分類することができます（これは，あくまで私の独断に基づいた分類です）．

- [A]　SDKのサンプルプログラムGENERICに出てくる関数
- [B]　初心者がまず最初に覚える必要のある関数
- [C]　ある程度使い込めば出てくるごくふつうのレベルの関数
- [D]　ほとんど使わない関数
- [E]　使ってはいけない関数

ランクAは，最初に参照するサンプルプログラムGENERICに登場するAPIです．ランクBは，これに続いて初心者でもよく使うと思われるものです．ここまでは，ぜひとも覚えておきたいものです．ランクCは，アプリケーションの処理の内容に応じて，必要となってくるものですので，これは，必要なときに覚えていけばよいでしょう．D，Eはあまり覚える必要のないものです．これだけの目安でも，ずいぶん違いがでてくるでしょう．

ここでは，ランクAからCまでに分類した関数を中心として，とくによく使われるものや，Windowsプログラミングの特徴である「オブジェクト指向的な考え方」が反映されているものについて解説していきます．

メッセージ関数

Windowsプログラムは，メッセージ駆動型だといわれます．Windowsシステムでは，マウスの動きや，キーボードからの入力，どのメニューが選択されたかなど，すべての事象がメッセージによってウィンドウに通知され，メッセージの内容によってプログラムが動作します．メッセージ関数は，メッセージをほかのウィンドウやアプリケーションキューに渡したり，メッセージを受け取ったりといった，Windowsシステムを語る上で欠かせない機能をもった関数です．

WinMain ［A］

――Windowsのmain関数

WinMainは，Cプログラムのmain関数に相当します．WinMainを抜けるとアプリケーションは終了します．ただし，Windowsアプリケーションでは，自分で確保したメモリやリソースは，自分で解放しておかなければ，ほかのアプリケーションがそれを利用できません．

それから，ふつうのメッセージ処理は，ウィンドウプロシージャの中で行っていますが，実はWinMainの中でもメッセージ処理ができるのです．『プ

FIG.1

Windows APIの種類（1）

- Windows API
  - ウィンドウズマネージャインターフェース関数
  - グラフィックデバイスインターフェイス関数
  - システムサービスインターフェイス関数

- ウィンドウズマネージャインターフェイス関数
  - メッセージ関数
  - ウインドウ作成関数
  - 表示と移動関数
  - 入力関数
  - ハードウェア関数
  - 描画関数
  - ダイアログ関数
  - スクロール関数
  - メニュー関数
  - 情報関数
  - システム関数
  - クリップボード関数
  - エラー関数
  - キャレット関数
  - カーソル関数
  - フック関数
  - プロパティ関数
  - 矩形関数

ログラミングWINDOWS』(アスキー刊)の「永遠に描画し続けるプログラム」でも，この方法を使っています．何らかの処理(『プログラミングWindows』の例では，さまざまな四角形を描画する)をWinMainの中に書いてしまうのです．何となくWindowsの仕組みが見えてくるような気がしませんか？

GetMessage [A]

――getcharと比較してみよう

GetMessageは，Cのランタイム関数のgetcharと非常によく似ています．getcharという関数は，キーボードから1文字入力されたら，リターン値としてアスキーコードを返します．つまり，getchar関数を実行しても，そのままでは何も値はもどってきません．キーボードが押されたら，そのときはじめて値が返るのです．GetMessageは，ちょうどキーボードが押されるとアスキーコードが返ってくるがごとく，メッセージがくるまで待ち続けます．待っている間は時間が無駄なので，ほかのアプリケーションが動くことができます．

リスト1は，getcharの例です．ここではキーボードから「Q」という文字が入力されるまでは{}内の処理を行います．{}の中には，通常，ほかの文字が入力された場合の処理などを記述することになります．リスト2は，GetMessage関数の例です．GetMessageでは，WM_QUITメッセージが送られてくると，リスト1で「Q」が送られてきたのと同じことになります．

DispatchMessage [A]

――Windowsさん，私を呼んで

初心者は，DispatchMessage関数とふつうの関数との違いがなかなかわ

COLUMN

オンラインヘルプを活用しよう

どんな時にどのAPIを使えばよいのか，この関数の引数は何だったか，そんなときにはオンラインヘルプを活用しましょう．SDKのヘルプを使うには次のようにします．

①Windowsからプログラムマネージャを起動し，ヘルプファイルSDKWIN.HLPを登録したいグループをオープンする．

②ファイル／新規登録メニューを選択，「プログラムを登録する」を選んで，「コマンドライン：」にSDKWIN.HLPのパス名を入力する．

③続いて「アイコン変更」ボタンをクリックし，「ファイル名：」に'WINHELP.EXE'と入力する．これで，Windowsのヘルプと同じアイコンに変更される．

④ファイルマネージャを起動し，SDKWIN.HLPファイルを反転させてから，ファイル／関連付けメニューを選択してWINHELP.EXEと関連づける．

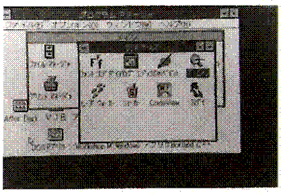
⑤SDKWIN.HLPのアイコンをクリックすると，このファイルを読み込んでオンラインヘルプが起動する．

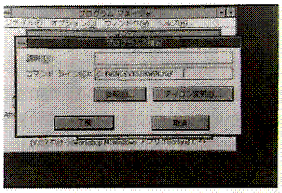
⑥「検索」ボタンを押して，検索画面を表示．

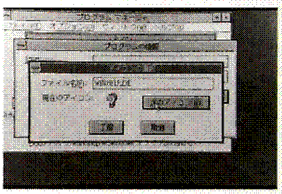
⑦検索したい関数名を入力して，リターンキーをすばやく2回押すと，関数の詳細が表示される．

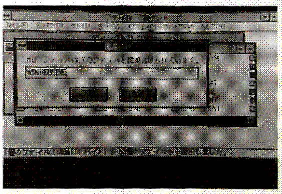
私自身はプログラミング時にオンラインヘルプとProgrammer's Page 2月号の綴じ込み付録「MS-Windows 3.0関数一覧」を併用しています．関数一覧で，関数名を探してからオンラインヘルプでその詳細を見るわけです．もっとぜいたくができるなら，開発マシン以外にもう1台Windowsの動くマシンを用意しましょう．

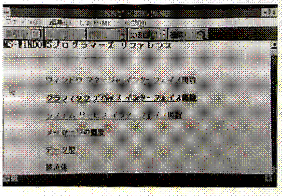
注：Programmer's Page 2月号はすでに品切れとなっております．今回の特集に際して，付録ディスクにMS-Windows 3.0関数一覧(MS-DOSテキストファイル)を収録いたしましたのでご利用ください．

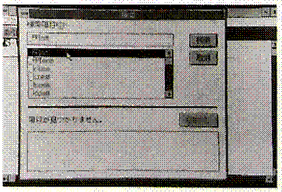

写真①

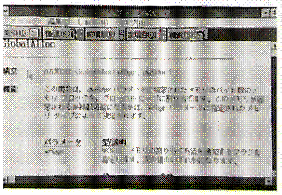

写真②


写真③


写真④


写真⑤


写真⑥


写真⑦

Windowsソフトウェア一覧
A&L FileChange，30,000円，図形データ変換ユーティリティ，ノックスインターナショナル

CHART.1
メッセージ関数

ランク	関数名	内容
C	CallWindowProc	メッセージの情報を指定した関数へ渡す
A	DispatchMessage	指定したウィンドウ関数へメッセージを渡す
A	GetMessage	指定した範囲からメッセージを取得する
D	GetMessagePos	最後にメッセージを取得したときのマウスの座標を返す
D	GetMessageTime	最後にメッセージを取得したときの時刻を返す
D	InSendMessage	現在のウィンドウ関数が, SendMessage 関数で受け取ったメッセージを処理しているかを調べる
C	PeekMessage	アプリケーションキュー内のメッセージを調べる
D	PostAppMessage	アプリケーションキューへメッセージを送る
B	PostMessage	アプリケーションキューにメッセージを入れる
A	PostQuitMessage	アプリケーションへ WM_QUIT メッセージを送る
D	ReplyMessage	メッセージに応答する
B	SendMessage	ウィンドウにメッセージを送る
D	SetMessageQueue	異なるサイズの新しいメッセージキューを作成する
C	TranslateAccelerator	メニューコマンドのためにアクセレータキーを処理する
C	TranslateMDISysAccel	マルチドキュメントインターフェースの子ウィンドウコマンドアクセレータを処理する
A	TranslateMessage	仮想メッセージを文字メッセージに変換する
D	WaitMessage	他のアプリケーションに制御を渡す
A	WinMain	Windows アプリケーションを実行するために, エントリポイントを提供する

からないでしょう. ウィンドウプロシージャを自分の関数内から呼べばよい, しかもこの時点ではどのウィンドウにどんなメッセージがあるのかは, メッセージ構造体の中に入っていてわかるのに, あえて DispatchMessage関数を使うのはなぜなのか.

これは, そもそも Windowsが構造上そうなっているから, としかいいようがありません. この処理を自分で全部書くとしたら, 相当なプログラムステップが必要になります. この時点で, どんなウィンドウがあるのか, メッセージはどのウィンドウにいくのか, それを自分で管理するよりも, Windowsシステムに「メッセージを渡してください」と, お願いするほうがいいのではないでしょうか.

Windowsシステムの構造として, 関数は Windowsから呼ばれるのが前提です. それぞれの API も, DLL としていろいろな関数から呼ばれますし, いつでもどこからでも呼ばれるというのが Windowsの基本構造なのです. それから, Windowsから呼ばれる可能性のある関数は, すべて long FAR PASCAL型にするというのもたいせつなポイントです.

PostMessage と SendMessage [B]
——送信のタイミングの違いに注意

PostMessage, SendMessage は, アプリケーションが自分自身でメッセージを送りたいときに使います. 使い方としては, おもに次の3つがあります.

1. 自分自身に送る
2. ほかのアプリケーションに送る
3. Windowsに送る

自分自身にメッセージを送る場合, メッセージ処理を利用していろいろとおもしろいことができます. 新しいメッセージを作ることもできるし, もともとあるメッセージを複数のウィンドウに送ることもできます. たとえば, あるゲームのプログラムの画面を考えてみてください. メインのシューティングの画面のほかに, 得点を表示するチャイルドウィンドウがあるとしましょう. そのウィンドウ上でマウスのボタンを押すと, これに対して WM_LBUTTONDOWN というメッセージが発生します. でも, このウィンドウ上でボタンが押されたという状況は, 背景のウィンドウにも伝えてやりたい. こういうときには, 自分がいま受け取ったのと同一のメッセージを, PostMessageで背景のウィンドウに向けて送りなおしてやればいいのです. こ

LIST.1

```
while ((c = getchar()) != 'Q'){
        :
        :
}
```

LIST.2

```
while ( GetMessage (&msg,       /* WM_QUITメッセージを受け取ったとき  */
         NULL,                  /* のみ, while文は偽の値を返す        */
         NULL,
         NULL)) {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
}
```

うすれば，画面上でボタンを押したという処理を両方のウィンドウに作る必要はありません．

自分自身にメッセージを送るもうひとつの例として，関数の代わりに使うという使い方もあります．ある処理(仮にAprocとします）を関数Aprocとして，内部に記述するというのが一般的なCプログラムの書き方ですが，Windowsでは，メッセージ処理を利用することもできます．WM_APROCという新しいメッセージを作って，これを自分自身に向けて送り，WM_APROCに対応する処理をウィンドウ手続きの中に入れるわけです．

では，内部的に関数として記述した場合と，メッセージ処理を利用する場合は，どのような違いがあるのでしょうか．メッセージ処理を利用する場合は，これをほかのプログラムが利用することもできます．つまり，WM_APROCというメッセージを送るのは，ほかのプログラムでもかまわないのです．これを利用して，たとえば，花札ゲームの中からゴルフゲームに，「ショットする」というメッセージを送ることもできないわけではありません．

Windowsで使われるメッセージは，メッセージ番号，wordパラメータ，longパラメータで構成されます．ユーザーが自由に設定できるメッセージ番号は，WM_USER(0x0400)から0x7FFFまでと決められています．したがって，新しいメッセージを作りたいときは，次のように定義してください．

```
#define  WM_APROC  WM_USER+1
```

メッセージ処理でもうひとつ挙げておきたいのは，PostMessageとSendMessageの違いについてです．ひとことでいうと，PostMessageは全部終わってから呼び出す．SendMessageはすぐに呼び出すということです（図2）．『ウインドウズ・プログラミング』（富士ソフトウェア刊）では，両者の違いについて，「PostMessageは同期で送り，SendMessageは非同期で送る」と説明されていますが，この表現だけではピンとこないでしょう．

これを，もう少し詳しくしたのが図3です．SendMessageは，あるウィンドウに対して直接メッセージを送りますから，処理aを実行した後，すぐにそのウィンドウのウィンドウプロシージャに処理が移ります．ウィンドウプロシージャでのメッセージ処理が終わると，制御がもどって処理bが実行されます．ところが，PoatMessageでは処理aの後では，メッセージキューに，あるウィンドウあてのメッセージが送られるだけです．処理bが終わって，メッセージの1サイクルが終わり，次のメッセージ処理のサイクルになっ

FIG.2

PostMessageとSendMessageの違い ①

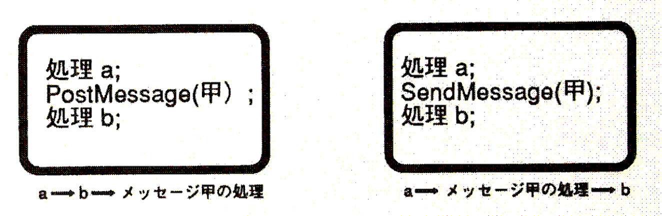
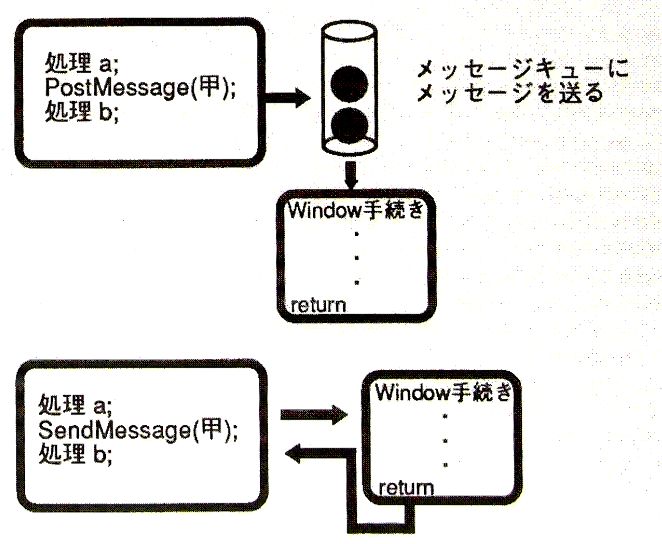


FIG.3

PostMessageとSendMessageの違い ②



Windows ソフトウェア一覧
Actor 3.0（英語版），188,000円，開発環境構築ツール，データポット

たとき，はじめてそのウィンドウプロシージャによってメッセージが処理されます．また，メッセージを送るウィンドウは，自分自身の場合もありますし，ダイアログボックスやコントロールの場合もあります．

この2つの関数はなかなかの曲者で，きちんと区別して理解しないと，自分の思った通りにプログラムが動作しません．エディットコントロール（これもひとつのウィンドウです）に何か文字を入れるというのも，ゴルフゲームでボールをショットするというのも，価値としては同等です．でも，エディットコントロールに文字を入れる場合は，文字が入ってから，はじめて次の作業に移れるわけですから，SendMessageを使わなければなりません．

その動作が直ちに行われるのか，それとも一通り処理が終わってから行われるのかという違いは，そのほかの関数でも十分に意識する必要があります．たとえば，PostQuitMessage関数を実行したのに，どうしてその場で処理が終わらないのだろう？　という疑問につきあたることになります．PostQuitMessageは，WM_QUITメッセージをPostする関数ですから，当然すべての処理を実行した後で，WM_QUITメッセージが処理されることになるのです．

PeekMessage［C］

――独自のメッセージループをつくる

PeekMessageは，プリンタまわりなどでよく使われます．また，アイコンの中で地球が回転するようなアプリケーションなどでも使われています．PeekMessageは，GetMessageと同じように，メッセージを取得する関数ですが，GetMessageと違って，待つ動作をしません．そして，ほかのアプリケーションのキューが空のときにも呼ばれます．つまり，誰にも迷惑をかけることなく，アイコンが変化するような遊び心のあるプログラムができるのです．

プリンタまわりでは，中止関数というのがあります．アプリケーションから印刷のコマンドを選択して了解ボタンをクリックすると，ダイアログボックスが出てきて中断したいときにクリックするボタンが表示されます．このような場合は，実はアプリケーション側が独自のメッセージループを持っています．その場合も，PeekMessageを使います．

CallWindowProc［C］

――サブクラスには必須

Windowsには，「サブクラス」という考え方があり，それ自体がWindowsのオブジェクト指向的な側面をよく表しています．

サブクラスを使う例として，Windows上で動作する英文ワープロ考えてみましょう．このワープロは，本来2バイト文字をサポートしていないものとします．そこで，日本語入力を可能にするために，もうひとつ別の簡単なアプリケーションを作成することにします．新しいアプリケーションは，入力される文字だけをチェックして，それが日本語の2バイト文字だったら，その内容に手を加えるという処理を行って，元の英文ワープロにもう一度送ってやるだけの処理をします．そうすれば，英文ワープロのソースコードの内容がわからなくても，処理の内容を変更できることになります（これは理論上の話にすぎません，実用的なワープロにするためには文字の入力だけでなく，そのほかのさまざまな点で変更を加えなければならないのはもちろんです）．

Windowsは，メッセージ駆動型のプログラムです．ふつうなら，アプリケーションプログラムは自分が主体になって動きますが，Windowsでは「メッセージ」という一種のメモの内容によって動作が決まります．そのメモを途中で誰かが横取りしたり，誰かが中身をこっそり書き換えたとしても，やはりその受け取ったメモの内容に従って処理を行うのです．このようなサブクラスを使った処理で，メッセージを横取りしたプログラムが，本来そのメッセージを処理すべき関数に渡すときにはCallWindowProc関数を使います．実際に利用する頻度は少ないかもしれませんが，用途によってはたいへん役にたちます．

ウィンドウ作成関数

ウィンドウ作成関数は，ウィンドウを作成する関数，いったん作成したウィンドウに関する情報を得る関数，ウィンドウの情報を変更する関数，ウィンドウを廃棄する関数などの集まりです．

DefWindowProc［A］

――SDKにはソースが入っている

DefWindowProcは，オブジェクト指向プログラミングの情報を隠ぺいする役割を果たしています．通常は，Windowsシステムというブラックボックスの中にメッセージが入っていく，これを奪ってきて何か処理をすると，このブラックボックスに新しい機能が加わるというのが，Windowsプログラミングの基本の考え方です．ブラックボックス内部の処理は，知らなくてもかまわないのです．Windowsでのブラックボックスの最たるものがDefWindowProcです．アプリケーシ

Windowsソフトウェア一覧
Ami PROFESSIONAL（英語版），99,000円，英文ワープロ，ジスクソフト

ョンプログラムで処理しないメッセージは，すべてDefWindowProcにまかせてしまいます．

ところで，SDKの中にはこのDefWindowProcのソースリストが入っています．それを見れば，何がどんな風に処理されているのかすべてわかってしまいます．入っているものを想像することができても，何が入っていないのかを判断するのは難しいものですが，これさえ見れば一目瞭然です．通常のインストールで，¥WINDEV¥SAMPLES¥DEFPROCS¥にあるDEFWND.Cというファイルがこれにあたります．

AdjustWindowRect［C］

——ウィンドウサイズを計算してくれる

CreateWindowを実行するとき，パラメータとして表示するウィンドウの縦横の長さを指定します．ところが，そこで指定する長さは，キャプションやメニューバーを含んだものなのです．もし，これこれの大きさのクライアント領域を作りたいというときには，非常に困ります．そこで，クライアント領域の幅と高さをAdjustWindowRectにセットすると，CreateWindowで指定する値を算出してくれるというわけです．

ただし，この関数はメニューが2段になる場合には使えません．

DestroyWindow［C］

——システムリソースを確保するために

あるウィンドウを開いて，そこにボタンをつけるとしましょう．このボタンは，たいへんメモリを消費します．プログラムマネージャの「ヘルプ」メニューから「Windowsについて」を選択すると，「使用可能システム リソース……xx %」と表示されますね．ボタ

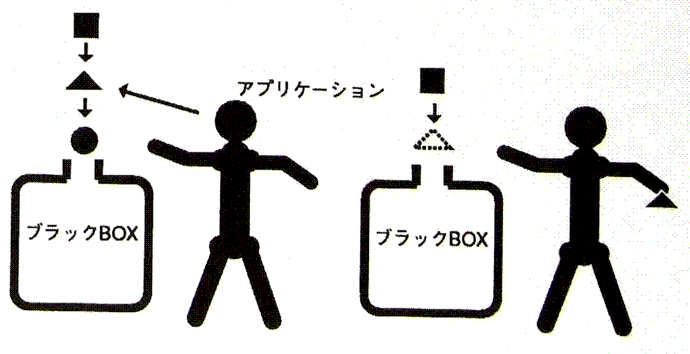
CHART.2

ウィンドウ作成関数

ランク	関数名	内容
C	AdjustWindowRect	あたえられたクライアント領域と一致するウィンドウサイズを計算する
C	AdjustWindowRectEx	あたえられたクライアント領域と一致するように，拡張スタイルのウィンドウサイズを計算する
A	CreateWindow	オーバーラップウィンドウ，ポップアップウィンドウ，子ウィンドウを作成する
C	CreateWindowEx	拡張スタイルのオーバーラップウィンドウ，ポップアップウィンドウ，子ウィンドウを作成する
C	DefDlgProc	アプリケーションが処理しないダイアログボックスメッセージのデフォルトの処理をする
C	DefFrameProc	アプリケーションが処理しない MDI のフレームウィンドウメッセージのデフォルトの処理をする
C	DefMDIChildProc	アプリケーションが処理しない MDI の子ウィンドウメッセージのデフォルトの処理をする
A	DefWindowProc	アプリケーションが処理しないデフォルトのメッセージ処理を用意する
C	DestroyWindow	WM_DESTROY メッセージを送出する
C	GetClassInfo	指定したクラスについての情報を取得する
C	GetClassLong	ウィンドウクラス情報を WNDCLASS 構造体から取得する
C	GetClassName	ウィンドウクラス名を取得する
C	GetClassWord	ウィンドウクラス情報を WNDCLASS 構造体から取得する
C	GetLastActivePopup	別ウィンドウが所有するポップアップウィンドウのうちで，最後までアクティブだったものを取得する
C	GetWindowLong	ウィンドウについての情報を取得する
C	GetWindowWord	ウィンドウについての情報を取得する
C	RegisterClass	ウィンドウクラスを登録する
C	SetClassLong	WNDCLASS 構造体の情報を置き換える
C	SetClassWord	WNDCLASS 構造体の情報を置き換える
C	SetWindowLong	ウィンドウの属性を変更する
C	SetWindowWord	ウィンドウの属性を変更する
C	UnregisterClass	ウィンドウクラステーブルからウィンドウクラスを削除する

FIG.4

Windows はブラックボックス



1　アプリケーションは自分の必要なメッセージを見つけると

2　それを横取りして処理を行います。

Windows ソフトウェア一覧
Btrieve，160,000円，データベース，エージーテック

ンひとつ作成するごとに，システムリソースが減少します．こういったものは，必要なとき作ればよく，用がなくなったらさっさと消してしまいます．ダイアログボックスにボタンをつける場合には，メモリの確保や廃棄はWindowsが自動的に行ってくれますが，自分で作ったウィンドウなどの場合は，自分で消去しなければなりません．このときDestroyWindowを使います．

Get××~Set××［C］

——2つをペアで覚えよう

GetClass××という関数は，RegisterClassで登録したクラスの情報を取得するときに使用します．GetWindow××という関数は，CreateWindowを実行したときにセットしたウィンドウの情報を取得するときに使用します．SetClass××，SetWindow××という関数は，これらの情報を変更するときに使用します．

Windowsでは，このように「値を最初に設定する関数」「情報を得る関数」「値を変更する関数」の3種類がセットになっていることが多いのです．RegisterClass，CreateWindowは，値を最初に設定する関数で，これに対して情報を得る関数が，GetClass××，GetWindow××，値を変更する関数がSetClass××，SetWindow××になります．

また，このように3種類の関数が別々に用意されているということ自体，Windowsのオブジェクト指向的な側面を表しているといえます．通常のCプログラム風に記述するならば，3種類も関数を用意する必要はないはずです．クラスやウィンドウに関する情報を登録する処理は，表面上は現れないある構造体に値をセットしていることにほかなりません．ふつうは番地さえわかっていれば，その値に直接アクセスしたり，その内容を変更することは自由にできるはずです．

しかし，Windowsではそうしてはいけません．どの場面でも，用意された関数を使わなければなりません．これは，Windowsの基本的な事項ですから，よく覚えておいてください．また，ひとつの項目について，設定・取得・変更という3種類の関数があることを知っているだけで，APIの体系的な理解は，ずっと楽になるはずです．

表示と移動関数

このグループは，Windowsの状態についての関数です．ウィンドウには，「最大化」「通常」「アイコン」という3つの状態があります．オブジェクト指向プログラミングの特徴として，Windowsではそれぞれの状態を取り扱うために，別々の関数が用意されています．たとえば，ウィンドウをアイコン化するときは，CloseWindow，アイコンをオープンするときはOpneIconという具合です．

CHART.3
値の設定／情報取得／変更

値を最初に設定する関数	RegisterClass	CreateWindow
情報を取得する関数	GetClassLong GetClassWord	GetWindowLong GetWindowWord
値を変更する関数	SetClassLong SetClassWord	SetWindowLong SetWindowWord

CHART.4
表示と移動関数

ランク	関数名	内容
C	ArrangeIconicWindows	最小化，またはアイコン表示された子ウィンドウを並べる
C	BeginDeferWindowPos	DefWindowPos関数が使うメモリを初期化する
C	BringWindowToTop	指定したウィンドウをオーバーラップしたウィンドウ中最上位のウィンドウにする
C	CloseWindow	指定したウィンドウを閉じる，またはアイコン化する
C	DeferWindowPos	EndDeferWindowPos関数で移動またはサイズ変更されたウィンドウの位置を記録する
C	EndDeferWindowPos	DeferWindowPos関数が記録している情報に基づいて同時にいくつかのウィンドウの移動またはサイズを変更する
C	GetClientRect	ウィンドウのクライアント領域の座標をコピーする
C	GetWindowRect	ウィンドウ全体をコピーする
C	GetWindowText	ウィンドウのキャプションをコピーする
C	GetWindowTextLength	与えられたウィンドウのキャプションの長さを返す
C	IsIconic	ウィンドウがアイコンとして表示されているかを調べる
C	IsWindowVisible	ウィンドウが画面上に存在しているかを調べる
C	IsZoomed	ウィンドウが最大化表示されているかを調べる
B	MoveWindow	指定されたウィンドウにWM_SIZEメッセージを送る
C	OpenIcon	アイコン化されたウィンドウをオープンする
C	SetWindowPos	子ウィンドウやポップアップウィンドウのサイズ，位置，順序を変更する
C	SetWindowText	ウィンドウのキャプションを設定する
C	ShowOwnedPopups	すべてのウィンドウを表示または非表示にする
A	ShowWindow	与えられたウィンドウを表示するかまたは削除する

「Defer」という文字列のついた関数は，ウィンドウの順番や大きさを変えるのに，ひとつずつ書き換えていたのでは，画面がちらついてしまいますから，内部的な構造体を変更してから一気に描きなおす，そういった処理を行う関数です．

また，「Is」ではじまる関数は，情報の取得を行います．IsIconicはアイコン化されているか，IsWindowVisibleは画面上に表示されているかどうか，IsZoomedは最大化されているかどうかを知るための関数です．これらのウィンドウ状態を最初に設定するのが，ShowWindowです．

それ以外の関数は，ウィンドウのサイズに関わるものです．ウィンドウというのは，通常タイトル，メニュー，クライアント領域などから構成されています．ウィンドウの座標を得るのはGetWindowRect，クライアント領域の座標を得るのがGetClientRectです．その座標の値を変更するには，MoveWindow，SetWindowPosを使います．

ShowWindow [A]

——ウィンドウ表示の状態を変更

デフォルトでは，CreateWindowの引数に，WS_VISIBLEというウィンドウスタイルを設定しない限り，Windowsは画面に表示されません．そこで，その状態を変更して，画面に表示されるようにするのが，ShowWindow関数です．ふつうは，CreateWindow，ShowWindow(hWnd,nCmdShow)，UpdateWindowの順で3つセットで使われます．これは，「むかしむかし」ときたら，次は必ず「あるところに」「おじいさんとおばあさんが住んでおりました」と続くのと同じようなものだと思ってください．では，なぜここで最初からShowWindow(hWnd，SW_SHOW)を使わないのでしょう．

プログラムマネージャで，「アクセサリ」グループを選択し，[SHIFT]キーを押しながら，メモ帳のアイコンをクリックしてみてください．起動したメモ帳ははじめからアイコンとして表示されますね．WinMainの4番目の引数に，nCmdShowというのがありますが，ShowWindowの引数もこれと同じです．通常は，SW_SHOW(ウィンドウをアクティブにして，現在のサイズ・位置で表示する)ですが，[SHIFT]キーを押しながらクリックすると，別の値がセットされます．つまり，ユーザーの呼びかけに応じた値がnCmdShowに入っているのです．

入力関数

入力関数は，マウス，キーボード，タイマからの入力を制御する関数です．

KillTimer/SetTimer [B]

——チェックしないとハングの原因に

タイマに関しては，必ずタイマが使用できるかどうかを，SetTimer関数のリターン値が0でないかどうかをチェックすることがたいせつです．Windowsでは，全部で15個までしかタイマが使えません．このチェックを怠ると，不可解なハングをするプログラムになってしまいます．

FIG.5

GetWindowRectとGetclientRectでの得られる座標

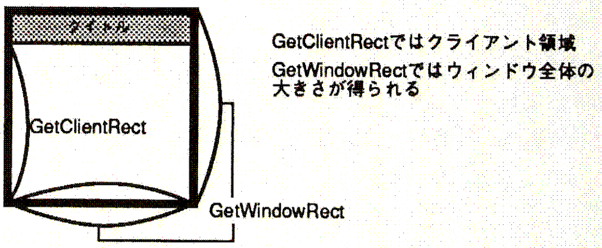


CHART.5

nCmdShowで表示されるウィンドウ状態

指定する値	意味
SW_HIDE	ウィンドウを非表示にする
SW_MINIMIZE	ウィンドウをアイコン表示にする
SW_RESTORE	SW_SHOWNORMALと同じ
SW_SHOW	ウィンドウをアクティブにして，現在のサイズ・位置で表示する
SW_SHOWMAXIMIZED	ウィンドウをアクティブにし，最大化ウィンドウとして表示する
SW_SHOWMINIMIZED	ウィンドウをアクティブにし，アイコン表示する
SW_SHOWMINNOACTIVATE	ウィンドウをアイコン表示にする，アクティブウィンドウは変更なし
SW_SHOWNA	ウィンドウを現在の状態で表示する，アクティブウィンドウは変更なし
SW_SHOWNOACTIVATE	ウィンドウを前と同じサイズ・位置で表示する，アクティブウィンドウは変更なし
SW_SHOWNORMAL	ウィンドウをアクティブにして表示する，ウィンドウがアイコン・最大化されているときは，元のサイズ・位置に復元

Windowsソフトウェア一覧
Click Art for WINDOWS3.0（全5巻），20,000円，グラフィックデータ集，レンガ

ハードウェア関数

ハードウェア関数は，通常のアプリケーションプログラムではほとんど使われませんが，ときには，GetAsyncKeyState，GetKeyStateなどが使われることがあります．

描画関数

Cプログラムでは，ファイルを取り扱うのにファイルポインタを使いますね．ファイルをオープンするときに，ファイルポインタを取得して，ファイルをクローズするときにもその値を使います．BeginPaint/EndPaint，GetDC/ReleaseDCでも，基本的な使い方はこれとよく似ています．Windowsでは，取り扱う内容がDCとなっていますが，このDCとはいったい何でしょうか？BASICのLINE文を考えてみてください．LINE文の引数には，線の始点と終点の座標，線の色，線の種類などを指定しますが，その引数が延々と続いたとしたらどうでしょう．わざわざ引数として指定するでしょうか．あるデータにその値をセットして，そのポインタだけを指定する方がずっとすっきりするでしょう．

DCとは，デバイスコンテキストのことで，実体は構造体です．しかし，直接それを操作してはいけません．どこかの番地に必ず存在するのですが，Windowsはそれを教えてくれません．その代わり，構造体の内容を取得・変更したりするための関数，Get/Setがあるのです．また，分類としてはGDI関数になりますが，CreateDC/DeleteDCという関数もあります．

デバイスコンテキストとは

CHART.6
描画関数

ランク	関数名	内容
A	BeginPaint	ウィンドウが描画するのを準備する
C	DrawFocusRect	フォーカスを表すスタイル内の矩形を描画する
C	DrawIcon	アイコンを描画する
B	DrawText	指定した文字列を描画する
A	EndPaint	ウィンドウの再描画の終わりをマークする
C	ExcludeUpdateRgn	ウィンドウ内の不正な領域に描かないようにする
B	FillRect	指定されたブラシを使って与えられた矩形を塗りつぶす
C	FrameRect	あたえられた矩形のために境界を描画する
B	GetDC	クライアント領域のディスプレイコンテキストを取得する
C	GetUpdateRect	ウィンドウの領域を境界の矩形の大きさだけコピーする
C	GetUpdateRgn	ウィンドウの更新領域をコピーする
C	GetWindowDC	ウィンドウ全体のディスプレイコンテキストを取得する
C	GrayString	淡色表示テキストで文字を書く
B	InvalidateRect	再描画のために矩形にマークをつける
C	InvalidateRgn	再描画のために領域にマークをつける
C	InvertRect	指定した矩形内のビットを反転表示する
B	ReleaseDC	ディスプレイコンテキストを解放する
A	UpdateWindow	ウィンドウの一部を描きなおす必要が生じたとき，アプリケーションに通知する
C	ValidateRect	再描画の指定から矩形を除外する
C	ValidateRgn	再描画の指定から領域を除外する

――実体は構造体

Windowsというシステムは，そもそも仮想的なもので，ディスプレイとプリンタ，プロッタには同じ価値しかありません．MS-DOSではリダイレクト機能があり，これを使えば標準出力であるディスプレイ表示を，ファイルやRS-232Cに変更できます．Windowsでも，まったく同じように，出力先を画面にするか，プリンタにするか，プロッタにするかを指定するだけなのです．画面に出力するときは，画面のデバイスコンテキスト（とくにディスプレイデバイスコンテキストと呼びます）を得ます．DCを得てから出力するまでの間に行う処理は，画面でも，プリンタでも，プロッタでも，まったく同じです．プリンタのDCを取得してから円を描く処理を実行すれば，プリンタに円が出力されるのです．

実際のところ，Windows自身もDCの中味はあまり見ていません．DCで示される構造体にセットされた値は，Windowsからデバイスドライバに渡されます．画面に表示したり，プリンタに印字したりするのは，デバイスドライバの処理になります．

BeginPaint/EndPaint ［A］
――WM_PAINTでは必ず使え

ディスプレイデバイスコンテキストを取り扱う関数として，BeginPaint/EndPaintとGetDC/ReleaseDCの2種類があります．前者はWM_PAINTメッセージのときに使うもの，後者はそれ以外のときに使うものです．誤って，WM_PAINTメッセージでGetDC/ReleaseDCを使ったりすると，たちどころにハングしてしまいます．WM_PAINTは，ちょっと変わったメッセージで，メッセージキューの中に浮かんだような状態になっています．他のメッセージがすべて処理されて，WM_PAINTがメッセージキュ

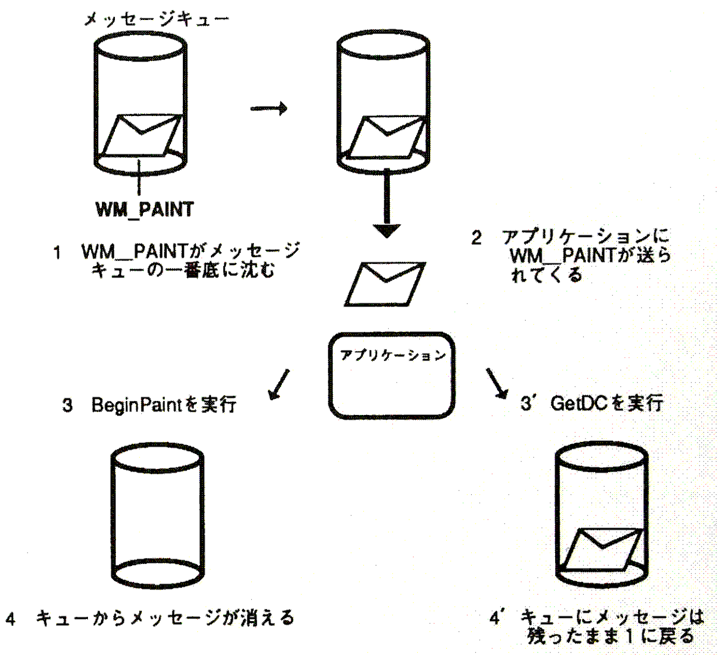
一の一番底に沈んでも，そのままではキューから落ちません．これを落とすのがBeginPaintの役割です．

WM_PAINTメッセージがキューの一番底に沈むと，メッセージのいわば分身が生まれて，これが送られ，処理されます（でも，メッセージキューにはWM_PAINTは残ったままです）．そして，BeginPaintを実行した瞬間，キューに残ったWM_PAINTが消えます．

もしここで，BeginPaintの代わりにGetDCを実行すると，メッセージは処理できますが，キューにはWM_PAINTが残ったままになります．次のサイクルでまたWM_PAINTメッセージを処理する，ここでGetDCを実行する……と，永遠に繰り返し，ハングアップします．たいせつなのは，WM_PAINTではBeginPaint/EndPaintを使うこと，GetDC/ReleaseDCを使ってはいけません．ただ単にブレークするだけというのもダメです．

FIG.6

BeginPaintとGetDCでのWM_PAINTメッセージ処理の違い



InvalidateRect［B］

――ウィンドウを描きなおす

Windowsの特徴として，すべてのウィンドウはアイコン化したり，バックグラウンドにしたりしても，再度画面に表示させると必ず元の状態を復元します．必ず直前の状態に再現できるように作らなくてはならないのがWindowsの掟なのです．この仕組みを利用すると，次のようなことができます．

簡単なメモ帳のアプリケーションを作成したとしましょう．現在，何行目から画面表示を開始するかという変数があります．仮に，いまこの変数をnとしましょう．このアプリケーションのウィンドウでは，現在一度に1行分表示できるとします．つまり，n行目が表示開始位置であるとき，これをアイコン化して，再度オープンすると，WM_PAINTメッセージが送られて，いま何行目が表示開始行かを判断し，n行目から画面を描きなおします（図7-a）．

今度は，画面を10行分上にスクロールすることを考えてみましょう．このときは，スクロールバーメッセージが送られてきます．この場合には，ScrollWindowという関数を使って，そのウィンドウに表示されているビットマップを移動するというやり方で描きなおすこともできます（図7-b）が，次のような方法もあります．

1．現在の表示開始行nに10を加える
2．自分自身に対してWM_PAINTメッセージを送る

これを表したのが図7-cです．Windowsでは，各ウィンドウは自分自身がどのような状態にあるかという情報を必ず保持していますから，その情報を利用すれば，新たな状態を作り出すことができるのです．このWM_PAINTを，自分自身に送るやり方のときに，InvalidateRectを使います．

初心者の方におすすめしたい使い方は，InvalidateRectの2番目の引数lpRectをNULLに，3番目の引数bEraseをTRUEにセットすることです．こうすれば，ウィンドウのクライアント領域全体が更新されますから，自分で持っている内部の変数を変えることによって，あらゆる描画ができるようになります．

ただし，全部を描きなおすのでは，画面がチラチラしてしまいますし，1

Windowsソフトウェア一覧
EXCEL2 プレタポルテ（パーソナルA），15,000円，Excel用フォーム＆データ集，マルテックス

行スクロールさせるだけなら，新たに描くのは最下行だけにするほうがスピードは速くなります．全体を描きなおすときには，内部で描きなおしても，WM_PANTを自分自身に送っても時間の違いはあまりありません．また，画面の再描画処理を一元的にやりたいというときには，このやり方が有効です．

ところで，InvalidateRect関数の説明を見ると「再描画のために矩形にマークをつける」とあります．「さて，いったいこれはどういう意味なんだろう？」ということになりますね．Windows自身も，よくこういった処理をしますが，画面を描きなおすとその部分だけ一瞬ですが，画面が真っ白になります．WindowsがWM_PAINTを送ります．DCにあるクリッピングエリア(この範囲にしか描画してはいけませんという領域です)を設定します．そうすると，このDCをBeginPaintが受け取って，指定されたクリッピングエリアだけを真っ白に塗りつぶします．その後，指定されたクリッピングエリアの内容を描きなおしているのです．

UpdateWindow ［A］
――いますぐ描きなおしたい

InvalidateRectは，WM_PAINTメッセージをSendではなくPostする命令です．したがって，「どうしてもいますぐ描きなおしたい」という時には不都合が生じます．たとえば，画面に丸を描いてその中に文字を表示したいとき，文字を先に書いて，その後から丸を描くためにInvalidateRectを使用すると，先に書いた文字は消えてしまいます．でも，うまい具合にPostをSendに代えてくれる命令があります．それがUpdateWindowです．UpdateWindowやShowWindowを使いこなせば，複数のウィンドウを使用するアプリケーションで，必要なタイミングで目的のウィンドウを表示したり，消去したりできるようになります．

ダイアログ関数

ダイアログ関数は，ダイアログボックスとコントロールの作成，変更，消去などを行う関数です．ダイアログ関数は，次の3つのグループに分類することができます．表7にそれぞれの関数をあげておきます．

①ダイアログボックスの作成・消去に必須の関数
②ダイアログボックス内のコントロールを制御する関数
③ダイアログボックス内の細かい操作を行う関数

その他のウィンドウマネージャインターフェース関数

ウィンドウマネージャインターフェース関数には，これまであげたもののほかに，スクロール関数，メニュー関

FIG.7

スクロールと画面表示

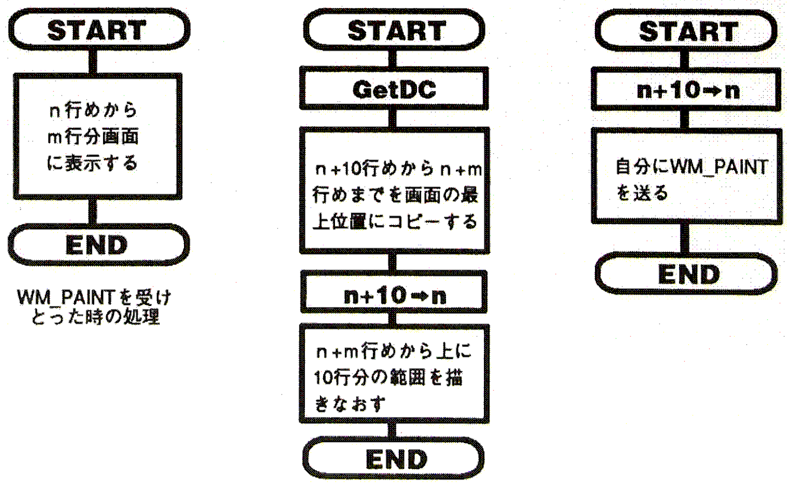


CHART.7

ダイアログ関数

作成・消去	
DialogBox	モードつきダイアログボックスを作成する
EndDialog	リソースを解析し，モードつきダイアログボックスに関連したウィンドウを消去する
コントロール制御	
CheckDlgButton	チェックをつける，取り除く，3つの状態をもつボタンの状態を変更する
ChckRadioButton	指定したボタンをチェックし，他のすべてのチェックを取り除く
DlgDirList	パスと一致するファイル名をリストボックスで一覧表示する
DlgDirListComboBox	コンボボックスにパスと一致するファイル名を一覧表示する
DlgDirSelect	リストボックスで現在選択されているものを取得する
DlgDirSelectComboBox	コンボボックスで現在選択されているものを取得する
GetDlgItemInt	あるコントロール内のテキストを整数値に変換する
GetDlgItemText	あるコントロール内のテキストを取得する
IsDlgButtonChecked	ボタンがチェックされているかを調べる
SendDlgItemMessate	指定されたコントロールにメッセージを送る
SetDlgItemInt	あるコントロールに指定された整数値を示す文字列をセットする
SetDlgItemText	あるコントロールに指定された文字列をセットする
その他の操作	
CreateDialog	モードなしダイアログボックスを作成する
CreateDialogIndirect	テンプレートからモードなしダイアログボックスを作成する
CreateDialogIndirect Param	テンプレートからモードなしダイアログボックスを作成し，データを渡す
CreateDialogParam	モードなしダイアログボックスを作成し，データを渡す
DefDlgProc	プライベートウィンドウクラスのダイアログボックスのデフォルトのメッセージ処理を行う
DialogBoxIndirect	テンプレートからモードつきダイアログボックスを作成する
DialogBoxIndirect Param	テンプレートからモードつきダイアログボックスを作成し，データを渡す
DialogBoxParam	モードつきダイアログボックスを作成し，データを渡す
GetDialogBaseUnits	ダイアログボックス作成時にWindowsが使ったダイアログベース単位を取得する
GetDlgCtrlID	子ウィンドウのIDを取得する
GetDlgItem	コントロールのハンドルを取得する
GetNextDlgGroupItem	コントロールグループ内の次(または前)のコントロールのハンドルを取得する
GetNextDlgTabItem	コントロールグループ内の次(または前)のコントロールでWS TABSTOPをもつもののハンドルを取得する
IsDialogMessage	メッセージが指定されたダイアログボックス宛のものかを調べる
MapDialogRect	ダイアログボックスの座標をクライアント座標に変換する

数，情報関数，システム関数，クリップボード関数，エラー関数，キャレット関数，カーソル関数，フック関数，プロパティ関数，矩形関数などがあります．そのほとんどがランクCないしランクDにあたります．これらの関数は，必要なときに応じて覚えていけばよいでしょう．関数名一覧とランクづけは，ここでは省略しますので，付録ディスク内のテキストファイルを参照してください．

グラフィックスデバイスインターフェース(GDI)関数

「デバイスコンテキスト」の項でも説明したように，Windowsは出力デバイスを問わない仮想的なシステムです．WindowsのGDI関数は，デバイスに依存しないでグラフィック操作を行う関数です．文字や図形，ビットマップを作成する関数，フォント操作やプリンタ制御の関数なども含まれます．

GDI関数については，SDKを使ってプログラミングをしていけば，いずれ出てくるランクCのものが中心ですが，ここでは，とくに注意すべき関数についてのみ取り上げます．

描画属性関数，マッピング関数
——DCの取得と変更

描画関数のところで説明した，デバイスコンテキストのGet/Setを行う関数群です．デバイスコンテキストのそれぞれの項目について，状態を取得する関数(Get×××)と状態を変更する関数(Set×××)が対になっています．

座標関数［C］
——座標系の変換

Windowsでは，さまざまな座標系が使われます．それぞれのアプリケーションが自分のウィンドウの中での位置を示す論理座標をもち，またディスプレイ上での絶対的な位置を示すための絶対座標も使われます．たとえば，グラフ表示アプリケーションの論理座標(10,10)に点を打つ場合，実際にその画面上の表示位置は，ピクセル単位の(50,50)に変換されるのです．このような，座標系の設定や変換を行ってくれるのが座標関数です．変換を行う関数があるのだということを知っておくのがポイントです．

カラーパレット関数［D］
——16色では使えない

カラーパレット関数は，256色表示が可能なマシンでのみ利用できます．16色表示しかできないマシンでは，カラーパレットを利用することはできま

せん．256色表示のハードウェアを備えている場合には利用できますが，NECのPC-9800シリーズをはじめとする国産の16色マシンでは，そもそもWindowsのディスプレイドライバ自体が256色表示に対応していないため，まったく利用できません．

システムサービスインターフェース関数

システムサービスインターフェース関数は，モジュール内のコードやデータへのアクセス，メモリの割り当て・管理，タスク管理，プログラムやリソースのロード，文字セットの変換，初期化ファイルの作成・変更，デバッグの支援など，各種システムサービスに関する関数です．

MakeProcInstance/FreeProcInstance［A］

FreeProcInstanceとMakeProcInstanceは，モジュールマネージャ関数に属します．Windows 2.0のときには，メモリ上の制限があったので，この2つの関数でさまざまな処理を行っていました．Windowsシステムでは，アプリケーションがメモリ上で，いつも同じ位置にロードされるとは限りません．アプリケーションの実行中も，同じコードがいつも同じ番地にあるとは限らないのです．そこで，内部にテーブルを設けて，システムがメモリの位置を変更するたびに，このテーブルを更新するようにし，アプリケーションプログラムは，ハンドルを使って各種メモリにアクセスするという間接的な方法をとります．

Windows 3.0では，プロテクトモードが利用できるようになったため，ハードウェアが肩代わりしてくれる部分が増えて，意味あいはやや軽くなってきています．とはいえ，Genericの

FIG.8
Windows APIの種類(2)

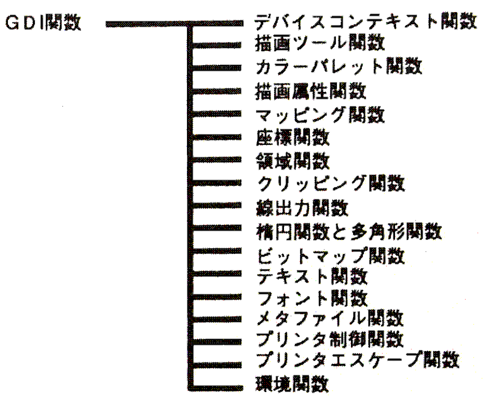
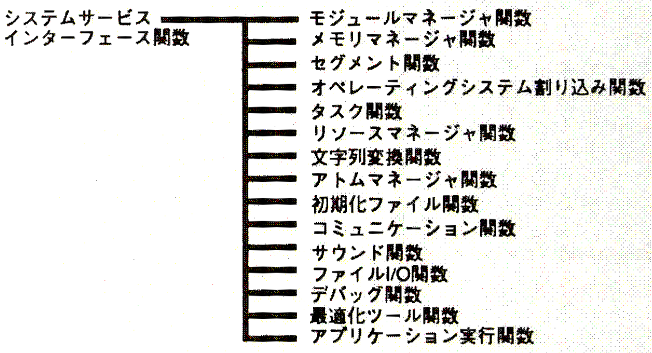


FIG.9
Windows APIの種類(3)



ようなアプリケーションでは，必ず使われる関数であることに変わりはありません．

初期化ファイル関数［B］

これは，アプリケーションで独自の．INIファイルを利用したり，WIN．INIファイルの内容を取得したり，値を書き込んだりする場合に使います．非常に便利な関数群です．

メモリマネージメント関数

メモリマネージメント関数には，グローバルメモリに対応したものと，ローカルメモリに対応したものが，それぞれ別にありますから，覚える内容は実際の関数の数の約半分です．詳しく説明すると，本が1冊書けるほどの内容がありますが，基本的な関数はメモリを確保したり，それを破棄したりするものです．

リソースマネージャ関数

各種リソース（ビットマップ，カーソル，アイコン，文字列）をロードする関数が含まれます．

文字列変換関数［C］

文字列の大文字・小文字変換や，文字列の比較・連結，文字列が英字，英数字，大文字，小文字であるかを調べるなど，文字列に関するユーティリティ関数群です．FARポインタに対応しているAPIであるため，DLLから呼び出されます．そのため，Cのランタイム関数を利用する場合と比べると，結果として実行ファイルのオブジェクトが小さくてすむ等のメリットがあります．

セグメント関数［E］

これらの関数は，後のバージョンとの互換性のため，使用しないよう推奨されています．

メニューとダイアログボックス

いちばん簡単な作り方　Windowsアプリケーションでメニューやダイアログボックスを使うには，リソース記述ファイルを作成する必要があります．しかし，リソースファイル中で定義したメニューを，アプリケーションプログラムで操作する（グレー表示にして選択できないようにしたり，チェックをつけたりする）ときにどうするかは，すぐにはピンとこないと思います．ここでは，メニューとダイアログボックスの一番簡単な作り方を紹介します．

メニューの作り方

ウィンドウアプリケーションの中でメニューを使うには，次の2つの手続きが必要です．

①テキストエディタを使って，リソース記述ファイルにメニュー定義を記述する．
②アプリケーションプログラム中で，ウィンドウクラス構造体にメニュー名をセットしてから，ウィンドウクラスを登録する．

ウィンドウでメニューを使えるようにするには，これだけでよいのです．メニューの表示や，選択，カーソル位置のメニューを反転表示させるなどの処理についてはWindowsに任せます．ユーザーがメニューを選択すると，ウィンドウにWM_COMMANDメッセージが送られます．Windowsは，どのメニューが選択されたかをウィンドウプロシージャの引数にセットしてくれますから，アプリケーションはそれを見て処理を行います．

しかし，いったん設定したメニューをグレーに変更したり，チェックマークを設定したりするには，さらに次の処理が必要になります．

③対応するメニュー関数を呼ぶ．

リソース記述ファイル

リスト3は，リソース記述ファイルのメニュー定義の例です．4行目のPpageMenuがこのメニューの名前となり，続くBEGIN～END間がメニュー内の項目定義になります．6行目のPOPUP文は，ポップアップメニューの定義の開始を示します．続くBEGIN～END間が，ポップアップメニュー「Help」を選択したときに表示されるメニューです．8行目のMENUITEM文は，表示するテキストとそのメニューを選択したときに返される結果を伴います．ここでは，「Help」メニューから「About Ppage」を選択すると，IDM_ABOUTという値が返ります．IDM_ABOUTの値は，ヘッダファイルPpage.hに定義されています．

メニュー名の登録

アプリケーションプログラム中では，ウィンドウのインスタンスを初期化する関数InitApplication中で，ウィンドウクラス構造体のlpszMenuNameに先ほどリソース記述ファイルで指定したメニュー名PpageMenuをセットします．RegisterClassでウィンドウクラスを登録すると，以後このクラスのウィンドウではPpageMenuが

CHART.8

ローカルメモリとグローバルメモリの関数対応表

グローバルメモリ用	ローカルメモリ用	機能
GrobalAlloc	LocalAlloc	ヒープからメモリを割り当てる
GrobalCompact	LocalCompact	コンパクトをかける
GrobalDiscard	LocalDiscard	ロックカウントが0ならメモリブロックを破棄する
GrobalFlags	LocalFlags	メモリブロックに関する情報を取得する
GrobalFree	LocalFree	メモリブロックを解放する
GrobalHandle	LocalHandle	オブジェクトのハンドルを調べる
GrobalLock	LocalLock	ロックカウントを増やしてメモリ中のブロックをロックする
GrobalReAlloc	LocalReAlloc	メモリブロックを再び割り当てる
GrobalSize	LocalSize	メモリブロックサイズをバイト単位で返す
GrobalUnlock	LocalUnlock	メモリブロックのロックを解除する

使えるようになります（リスト5）.

メニューをグレー表示にするには

いったん作成したメニューの特定の項目をグレー表示にするには，GetMenu関数でウィンドウのメニューのハンドルを取得し，EnableMenuItem関数を使って，指定したメニューの（hMenu）指定した項目（Item）について，メニューの状態（wEnable）を変更します．メニューの状態には，使用許可・使用禁止・グレー（淡色）表示のいずれか，トップレベルメニューかポップアップされたときに表示されるメニューかを指定できます．たとえば，リスト3のようにリソース記述ファイルに定義されたメニューに対して，リスト7を実行すると，「Help」がグレーで示され，リスト8を実行すると「About Ppage」がグレーで表示されます．

なお，メニューにチェックマークをつける場合には，CheckMenuItem関数を使いますが，その仕組みはEnableMenuItemとまったく同じです．チェックマークをつけるときは，MF_CHECKED，はずすときはMF_UNCHECKEDをセットします．

ダイアログボックスの作り方

Windowsのダイアログボックスには，モードつきとモードなしの2種類があります．モードつきダイアログボックスとは，そのダイアログボックスの処理を終えるまで親ウィンドウへ制御をもどせないもの，これに対してモードなしダイアログボックスは，表示中でも親ウィンドウへ自由に制御を移せるものです．

ここでは，先ほど作成したメニューで，「About Ppage」を選択したとき

LIST.3

ppage.rc

```
#include "windows.h"
#include "PPage.h"

PPageMenu MENU
BEGIN
     POPUP        "&Help"
     BEGIN
          MENUITEM "&About PPage...", IDM_ABOUT
     END
END

```

LIST.4

ppage.h

```
#define IDM_ABOUT 100

int PASCAL WinMain(HANDLE, HANDLE, LPSTR, int);
BOOL InitApp(HANDLE);
BOOL InitInst(HANDLE, int);
long FAR PASCAL WndProc(HWND, unsigned, WORD, LONG);
BOOL FAR PASCAL AboutProc(HWND, unsigned, WORD, LONG);

```

LIST.5

```
BOOL InitApp(hInstance)
HANDLE hInstance;
{
     WNDCLASS  wc;

     wc.style = NULL;
     wc.lpfnWndProc = WndProc;

     wc.cbClsExtra = 0;
     wc.cbWndExtra = 0;
     wc.hInstance = hInstance;
     wc.hIcon = LoadIcon(NULL, IDI_APPLICATION);
     wc.hCursor = LoadCursor(NULL, IDC_ARROW);
     wc.hbrBackground = GetStockObject(WHITE_BRUSH);
     wc.lpszMenuName =  "PPageMenu";
     wc.lpszClassName = "PPageWClass";


     return (RegisterClass(&wc));
}

```

LIST.6

```
HMENU hMenu;
hMenu = GetMenu(hWnd);
EnableMenuItem(hMenu, Item , wEnable);
```

LIST.7

```
EnableMenuItem(hMenu, 0, MF_BYPOSITION | MF_GRAYED);
```

LIST.8

```
EnableMenuItem(hMenu, IDM_ABOUT, MF_BYCOMMAND | MF_GRAYED);
```

CHART.9

メニューの状態－wEnable

メニューの表示形式	
MF_ENABLED	使用許可
MF_DISABLED	使用禁止
MF_GRAYED	グレー（淡色）表示
変更するメニュー	
MF_BYPOSITION	トップレベルのメニュー（左から0,1,2…）
MF_BYCOMMAND	ポップアップメニュー

に表示されるアバウトボックスを作ります．ボックス中のテキストコントロールには，このプログラムのバージョンを表示します．なお，アバウトボックスは，「了解」ボタンを押すまで親ウィンドウには戻れませんから，モードつきダイアログボックスになります．アバウトボックスの作成には次の手続きが必要です．

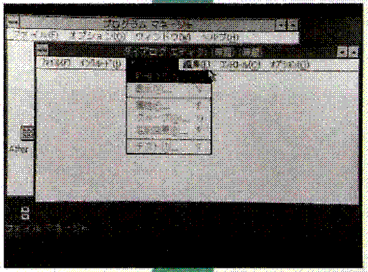
①ダイアログエディタを使って，ダイアログ記述ファイルとインクルードファイルを作成する．
②リソース記述ファイルにインクルードファイルとダイアログ記述ファイルをインクルードする．
③アプリケーションにダイアログボックスを操作する関数を記述する．

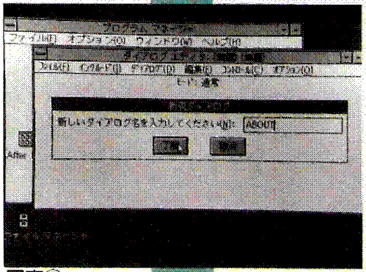
ダイアログエディタの使い方

ダイアログエディタは，ダイアログボックスの実際のイメージを画面で見ながら作成するツールです．

①ダイアログエディタを起動する．
②メニューから「ダイアログ／新規作成」を選択する．
③ダイアログ名をきいてくるので，ダイアログ名（ここではABOUT）を入力する．ボックスの外枠を示す四角が表示されるので，マウスで大きさを決める．
④メニューから「オプション／ツール」を選択する．
⑤道具箱の「T」（テキスト入力）をクリックする．カーソルを②の外枠の内側に移動して，テキストを表示させたい位置でクリックすると，Textの文字と文字枠が表示される．
⑥Textという文字をダブルクリックすると，詳細を設定する画面が表示されるので，「ID値」に「ID_VERSION」と入力して，「了解」ボタンをクリックする．
⑦インクルードファイルにID値の情報を追加するかどうかをきいてくるので，「了解」ボタンをクリックする．
⑧インクルードファイルの表示／編集画面が表示されるので，「追加」ボタン，続いて「了解」ボタンをクリックする．
⑨⑥の画面にもどるので，再度「了解」ボタンをクリックする．
⑩道具箱の「OK」をクリックして，カーソルを②の外枠の内側に移動し，ボタンを貼りつける．
⑪ボタンの中央をクリックすると，詳細を設定する画面が表示されるので，「ID値」に「1」テキストに「OK」を入力し，「了解」ボタンをクリックする．
⑫②の枠内でかつ⑤の文字枠，⑩のボタン以外の場所をダブルクリックすると，ダイアログボックスの詳細を設定する画面が表示されるので，「タイトル」に「About Ppage」を入力し，オプションから「WS_CAPTION」をチェック

写真①

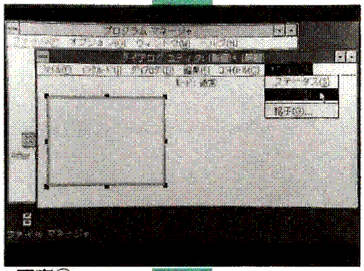

写真②


写真③

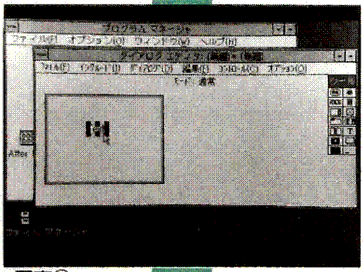
Windows ソフトウェア一覧
Hybrid ISAM，98,000円，C言語開発支援ツール，ハイドロシステム

して「了解」ボタンをクリックする．

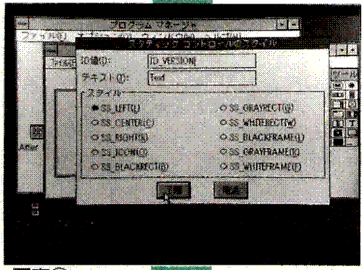
⑬「ファイル／新規保管」コマンドで作成したアバウトボックスを保管する．



写真④



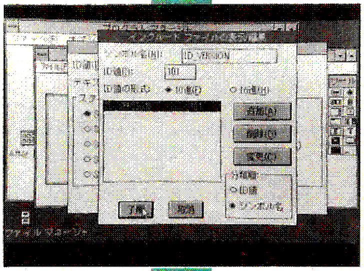
写真⑤



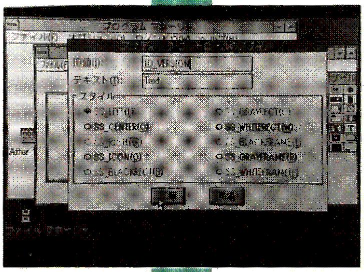
写真⑥

写真⑦

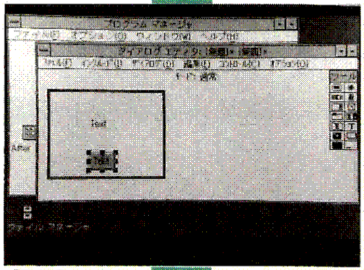
ダイアログエディタの作業を終えると，拡張子がRES, DLG, Hの3つのファイルができます．.RESは次にダイアログエディタで編集するときに参照するバイナリ情報，.DLGはリソース記述ファイルに追加するためのダイアログボックスのリソース記述を書いたテキスト情報（リスト9），.Hはリソース記述ファイルの先頭で定義するヘッダファイル（リスト10）です．これらのファイルをリソース記述ファイルに含めたものが，リスト11です．ヘッダファイルは，#inculde文，DLGファイルはrcinclude文でインクルードします．



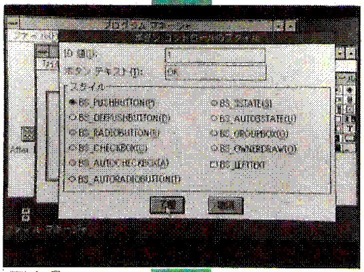
写真⑧



写真⑨



写真⑩



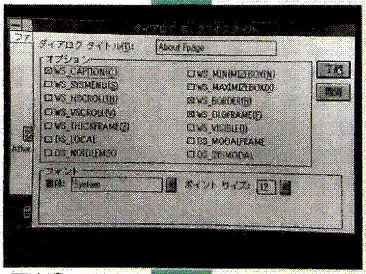
写真⑪

アプリケーションプログラムの記述

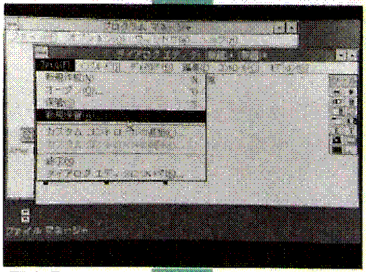
メニューから「About Ppage」が選択されると，ウィンドウプロシージャWndProcには，WM_COMMANDメッセージが送られて，引数wParamの値がIDM_ABOUTにセットされてきます．

このウィンドウ手続きでは，次の一連の処理を行います．

①MakeProcInstanceを実行して，ダイアログボックス処理を行う関



写真⑫



写真⑬

数Aboutへのポインタを取得する.

②ダイアログボックスのリソース名, ①のポインタなどを引数としてDialogBox関数を実行する.

③FreeProcInstanceを実行して, 関数を解放する.

実際には, ②のDialogBoxを実行すると, ダイアログボックスの実際の処理を行う関数AboutProcがWindowsから直接呼ばれます. ダイアログボックス処理関数では, WM_INITDIALOGという初期化メッセージ処理を記述します. このとき, 先ほどダイアログエディタで記述したID_VERSIONというIDに文字列「Version Beta」をセットします. 次にメッセージ処理として,「OK」ボタンが押されたときには, EndDialog関数でダイアログボックスを終了するようにします.

LIST.9

ABOUT.DLG

```
ABOUT DIALOG LOADONCALL MOVEABLE DISCARDABLE 9, 26, 216, 80
CAPTION "About Ppage"
STYLE WS_BORDER | WS_CAPTION | WS_DLGFRAME | WS_POPUP
BEGIN
    CONTROL "Text", ID_VERSION, "static", SS_LEFT | WS_CHILD, 58, 12, 69, 8
    CONTROL "OK", 1, "button", BS_PUSHBUTTON | WS_TABSTOP | WS_CHILD, 88, 60, 24, 14
END



```

LIST.10

ABOUT.H

```
#define ID_VERSION   101
```

LIST.11

ppage.rc

```
#include "windows.h"
#include "PPage.h"
#include "about.h"

PPageMenu MENU
BEGIN
    POPUP        "&Help"
    BEGIN
        MENUITEM "&About PPage...", IDM_ABOUT
    END
END

rcinclude about.dlg

```

LIST.12

```
/****************************************************************************
    Window message processing procedure
****************************************************************************/

long FAR PASCAL WndProc(hWnd, message, wParam, lParam)
HWND hWnd;
unsigned message;
WORD wParam;
LONG lParam;
{
    FARPROC lpProcAboutProc;

    switch (message) {
        case WM_COMMAND:
            if (wParam == IDM_ABOUT) {
                lpProcAboutProc = MakeProcInstance(AboutProc, hInst);

                DialogBox(hInst,
                    "ABOUT",
                    hWnd,
                    lpProcAboutProc);

                FreeProcInstance(lpProcAboutProc);
                break;
            }
            else
                return (DefWindowProc(hWnd, message, wParam, lParam));

        case WM_DESTROY:
            PostQuitMessage(0);
            break;

        default:
                return (DefWindowProc(hWnd, message, wParam, lParam));
    }
    return (NULL);
}


/****************************************************************************
    About box message processing procedure
****************************************************************************/

BOOL FAR PASCAL AboutProc(hDlg, message, wParam, lParam)
HWND hDlg;
unsigned message;
WORD wParam;
LONG lParam;
{
    switch (message) {
        case WM_INITDIALOG:
            SetDlgItemText(hDlg, ID_VERSION, "Version Beta");
            return (TRUE);

        case WM_COMMAND:
            if (wParam == IDOK
                || wParam == IDCANCEL) {
                EndDialog(hDlg, TRUE);
                return (TRUE);
            }
            break;
    }
    return (FALSE);
}

```


Windows ソフトウェア一覧
Im・Drive24, 15,000円, ディスプレイドライバ, 住友セメントシステム開発

特集 Windows プログラムの作り方

Windowsでも威力を発揮するオブジェクト指向言語 Borland C++

大野元久

長らく待望されていたBorland C++の日本語版がついに発売された．MS-C+SDKの2分の1のコストで，Windowsプログラミングに必要なすべてのツールが入手できるメリットは大きい．とくに，予算に余裕のないアマチュアプログラマにとっては朗報といえるだろう．Windowsアプリケーションにはなっていないものの，ボーランド社の十八番であるプログラミング統合環境（IDE）やTurbo Debuggerを備え，さらにWindows上でビジュアルにリソースを作成できるResource Workshopも付属している．また，統合環境版，コマンドライン版のそれぞれにプロテクトモードで動作するコンパイラが用意されているので，大規模なアプリケーションの開発も問題なく行える．
ここでは，このBorland C++の強力な開発環境を紹介するとともに，GUIプログラミングに威力を発揮するといわれているオブジェクト指向言語C++を用いたWindowsプログラムの作成法について解説する．

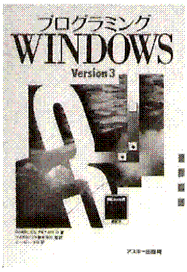
Borland C++の登場が，Windowsアプリケーションを作りたいと思っていたプログラマにとっての朗報であることはまちがいないでしょう．定評のあるTurbo C++やTurbo Debuggerと同様のユーザーインターフェースをもった環境が，Windowsアプリケーションの開発に利用できるのですから．さらにResource Workshopを使えば，これらと同様のユーザーインターフェースで視覚的に各種のリソースを編集できます．

現時点での日本におけるWindowsアプリケーションの数は，かなり少ないといわざるを得ません．もし，開発環境の不備がアプリケーション開発の妨げとなっているのだとしたら，Borland C++によってそれらの多くが改善されるのではないでしょうか．Cでのプログラミングで威力を発揮するのはもちろんですが，Windowsアプリケーション開発において威力を発揮するのは，やはりC++でのオブジェクト指向プログラミングです．

ここですべてを紹介することはできませんが，Borland C++でWindowsアプリケーションを開発する手順について解説します．この記事が，Windowsアプリケーションを開発するための参考になれば幸いです．

なお，CによるWindowsのプログラミングについては，ひととおり学習されていることを前提としています．残念ながら，Borland C++のマニュアルには，Windowsプログラミングの具体的な解説がほとんど記載されていません．Borland C++でWindowsアプリケーションを開発する場合には，『プログラミングWINDOWS』(Charles Petzold著，エー・ピー・ラボ訳，アスキー出版局）が必須となるでしょう．この原稿を執筆している時点では，日本語訳はWindows 2.0用のものしかありませんが，記載内容のほとんどは3.0でも通用します．プログラムについても，まったく変更しないか多少の変更ですむものばかりです．

プログラミング
Windows Ver.3



Borland C++の特徴

Borland C++の概要

Borland C++の統合開発環境(IDE)は，Programmer's Platformと呼ばれています．IDEは，従来のTurbo C++同様DOSアプリケーションであり，Windowsアプリケーションではありません．したがって，WindowsのGUI環境だけでWindowsアプリケーションを開発できるわけではありません．

しかし，WindowsからDOSアプリケーションとしてIDEを呼び出し，コンパイルすることはできます．一般に，WindowsからDOSアプリケーションを呼び出すと，メモリ不足が生じることが多いのですが，Borland C++にはXMS(プロテクトメモリ)を利用して大きなメモリ空間を利用するコンパイラが用意されていますから，プロテクトメモリを十分に備えていれば，まずメモリ不足になることはありません．WindowsとIDEを切り換えて使えば，DOS上のIDEの使い勝手をそのままWindowsアプリケーションの開発に応用することができます．

必要な環境

Borland C++のすべてのファイルをインストールするためには，ハードディスクの空き領域が20MB以上必要です．サンプルプログラムなどのインストールオプションをオフにすれば必要量を小さくすることはできますが，アプリケーションを開発するため（後述するプリコンパイルヘッダのため）には，十分な空き領域を確保しておくことが必須です．SDKは必要ないので，開発環境はこれですべてですが，ほかのアプリケーションを組み込むことを考えれば，最低でも80MB以上のハードディスクが欲しいところです．

Borland C++は，拡張メモリとしてEMSメモリとプロテクトメモリを使えますが，Windowsを快適に動作させるためにもプロテクトメモリの使用をお勧めします．できれば拡張スロットではなく，本体に内蔵できるタイプのものがよいでしょう．プロテクトメモリとともに，CPUパワーやマウスもBorland C++には必要です．もっと質素な環境で開発することもできますが，Windows用のアプリケーションを開発するのであれば，Windows自身を快適に利用するくらいの環境は整えておくべきです．

どのコンパイラを使うべきか

Borland C++には，コマンドライ

ンコンパイラ(BCC)と統合環境に組み込まれたコンパイラ(BC)のそれぞれについて，リアルモードで動作するものとプロテクトモードで動作するもの(コマンド名の後ろにXがつく)の2種類ずつ，計4種類のコンパイラが含まれています．CとC++という区別はありません．

これらのコンパイラは，開発環境によって使い分けることができます．仮想86EMSが組み込まれていれば，プロテクトモードコンパイラは使えませんから，リアルモードのコンパイラを使うことになります．使い慣れたエディタとコマンドラインコンパイラを組み合わせて使ったり，エディタやデバッガがすべて組み込まれた統合環境を使うことができます．

これら4種類のコンパイラの中で，Windowsアプリケーションを開発するのに最適なのは，統合環境を備え，さらにプロテクトモードで動作するBCX.EXEでしょう．

リアルモードで動作するBCはオーバーレイ領域としてプロテクトメモリを使うので，メインメモリはあくまでも640KBの範囲に限られます．このため，使用可能メモリが少なければオーバーレイ用のプロテクトメモリがどれだけあっても“Out of memory”というエラーを避けることはできません．またBCは，TC(Turbo C++の統合開発環境)よりも若干メモリの消費量が増えていますから，使用可能メモリが450～500KBくらい必要です．Windowsから DOSコマンドを呼び出すときの使用可能メモリは，多くても500KBくらいですから，かな漢字変換フロントエンドプロセッサ（FEP）を組み込んでおいて，Windowsから呼び出したBCでFEPを利用する，ということは難しいでしょう．

BCXは，プロテクトメモリをオーバーレイとしてではなくプロテクトモードで(つまり本来の形で)使います．DOSの使用可能メモリが200KBでも，プロテクトメモリが2MBあれば，統合環境でその2MBをメインメモリと同じように利用できるわけです．

ここで，なぜプロテクトモードを利用するコマンドラインコンパイラ(BCCX)を勧めないかというと，BCCXがプロテクトモードを利用するTLINKXではなくTLINKを呼び出しているからです．これは，プロテクトモードで動作するプログラムから別のプロテクトモードで動作するプログラムを呼び出せないためなのですが，TLINKの制限はやはりDOSの使用可能メモリに制約されてしまうためあまり好ましいとはいえません．この問題は，makeを使えば簡単に解決できますが，食わず嫌いをせずに一度統合環境を使ってみて欲しいと思います．使いなれると，これほど便利なものはないのですから．

BCXを起動する

Windowsから BCXを利用するときに，忘れてはならない注意事項を説明します．Windowsは，基本的にHIMEM.SYSがサポートしているメモリを全部Windowsのために使います．また，PIFファイルで子プロセスのために，XMSメモリを割り当てても BCXはそれを利用しません．BCXがプロテクトメモリを使うためには，BCXが利用するメモリをWindowsよりも先に確保しておかなければならないのです．これを行うのがTKERNEL.EXEです．

FIG.1

```
C>TKERNEL HI=YES KILOS=2048
```

TKERNELは，Windowsを起動する前に図1のようなコマンドラインを使って常駐させておきます．KILOS=2048は，BCXが使用できるプロテクトメモリの大きさです．この値を増やせばBCXはより多くのメモリを利用できますが，それだけWindowsの利用できるメモリは減ります．

Windowsはスタンダードモードで起動します(WIN /S)．エンハンストモードではDOSプロンプトが仮想86モードで動作するため，プロテクトモードを使うBCXが実行できないからです．

「ファイル(F)｜新規登録(N)」でBorlandグループ（グループ名はなんでもよい)を作成し，さらにプログラム

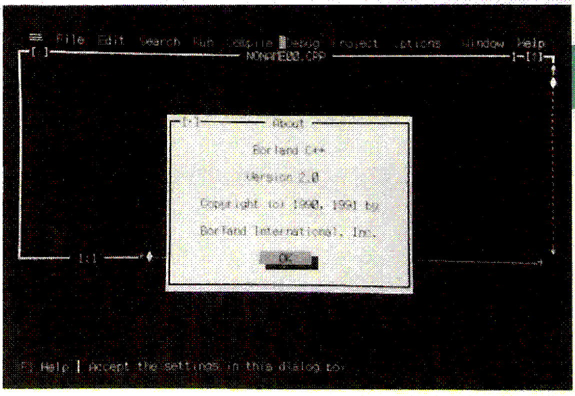


写真1
Borland C++の起動画面

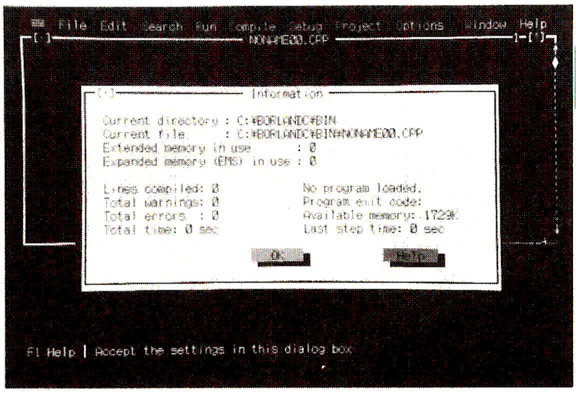


写真2

BCXから使用できるメモリ量

として BCX.EXE を登録します．ここでは「説明 (D)」にも BCX を指定するものとします．BCX は，Windows アプリケーションではないので，デフォルトのアイコンは，DOSプロンプトと同じそっけないものしか設定されません．「ファイル (F) | 登録情報 (P)」で，アイコンとして BC.ICO を指定すれば見栄えもよくなります．

H98ユーザーは，ほかにも注意しなければならないことがあります．H98用 MS-DOS を使って H98用 Windows をインストールしても，Windows は拡張アトリビュートに対応してくれません．試しにエンハンストモードで BC を呼び出し，DOSプロンプトをウィンドウモードにしてみてください．拡張アトリビュートは見る影もなく，通常アトリビュートになってハイライト文字だけがむなしく表示されます．

NECは，拡張アトリビュートを使うアプリケーションは，DOSプロンプトで使うようにアナウンスしているようですが，[CTRL]-[ESC]でタスクを切り換えるときの画面は不気味な背景色で塗りつぶされてしまいます（アトリビュートの上位バイトが不定になっているのでしょう）．おまけに，Windows アプリケーションからは，拡張アトリビュートに関する操作が認められていないようで，これを行うと Windows の画面が乱れてしまいます．

このためでしょうが，Turbo Debugger for Windows では，通常アトリビュートだけがサポートされています．

プリコンパイルヘッダ

Windows アプリケーションを開発するためには，必ず windows.h というヘッダファイルをインクルードしなければなりません．ところが，この windows.h というファイルは120 KB もあるのです．アプリケーションを構築するソースファイルが10KB しかなくても，このおきまりの120KB のコンパイルを避けることはできないのです．

しかし，それを律儀にコンパイルするのは，アプリケーション開発における時間の無駄です．Borland C++では，プリコンパイルヘッダという機能でこの無駄を省いています．つまり，ヘッダファイルをコンパイルした時点の内部情報を特別なファイルに保存し，再コンパイルの必要をなくしているわけです．windows.h のような，サイズが大きいだけで一度コンパイラを通してしまえばすむようなものには，プリコンパイルヘッダが特に有効です．ただし，プリコンパイルヘッダの情報を格納するファイルは，数100KB にも及びますので，ディスクの空き領域が少ない場合は使わないほうがよいでしょう．

Borland C++による Windows プログラミング

C++とオブジェクト指向について

多くの書籍や雑誌の記事が，C++やオブジェクト指向について解説していますので，あまり深くは立ち入りませんが，オブジェクト指向についてピンとこないという方のために少しだけ解説します．

一般に，扱わなければならないデータは，単独の数値や記号では表せないものです．たとえば，人の名前を並べてもそれだけでは何の役にもたちません．その人の住所や電話番号など，付随するデータが備わってはじめて住所録や電話帳として役立つものになります．C では，こうした性質をもつデータ型を「構造体」として定義していました．構造体のメンバは，データメンバ(メンバ変数)ですから，その性質は常にデータとしてのみ表されます．

しかし，あるデータ型を扱うというのであれば，それを処理するための「コード」(ここでは関数としておきます)が存在するはずです．C では，どの処理を行う関数も同じレベルでバラバラに存在しました．構造体は，それを扱う関数にどのようなものがあるか知りません．しかし，C++ではそうした関数を構造体の中に埋め込んでしまうことができます．この関数は，構造体のメンバとして定義されるのでメンバ関数と呼ばれます．いま，構造体は単なるデータ構造だけではなく，自分がな

Windows ソフトウェア一覧
Micrografx CLIPARTスタンダードエディション（英語版），30,000円，グラフィックデータ集，ジスクソフト

にをされるか，あるいはなにをすべきかという性質を知っています．そして，それをクラスと呼んでいるのです．

クラスは，ある性質を示す「型」として，その性質がどんなデータでどんな処理を行うのかを知っています.多くの場合,データ部分は隠されており,外部からそれを見ることはできません.それを見るためには,それを見せてくれるメンバ関数を用意しなければなりません．

クラスは型でしかありませんから，それを利用するにはその型の「実体」を定義しなければなりません．簡単にいえば変数のことです．この実体は，普通の変数と同様に定義することができますが，定義された実体がデータだけでなく，自分に関する処理も知っているということはすでに述べたとおりです．これが，オブジェクトあるいはインスタンスと呼ばれるものです．

C++では，継承という機能を使って既存のクラスの性質を拡張したり，2つの性質をもつクラスを組み合わせたりすることができます．定義済みのクラスの性質では，機能が不足するという場合でも，もとのクラスを拡張してもとのクラスに手を加えずに機能を拡張できます．クラスを継承して新しいクラスを作ることを，基本クラスから派生クラスを派生するといいます．Cでは，ある構造体に含まれているデータが不足していれば，その構造体自身を書き換えるか，別の構造体を用意しなければならなかったわけです．これが，C++が資産継承に役立つといわれる由縁です．

オブジェクト指向言語が，とくにWindowsプログラミングに適しているといわれるのは，ウィンドウをまさにオブジェクトと考えることができるからです．ウィンドウは，データと処理すべきことを知っている性質をもったオブジェクトなのです．Windowsは，アプリケーションが作成したウィンドウのためのウィンドウ関数にメッセージを送ります.C++では，ウィンドウのためのオブジェクトにメッセージの処理方法，つまりウィンドウ関数を教えてやればよいわけです．C++のクラスを使って，ウィンドウの性質を定義することができます．さらに，そこから異なる性質をもったウィンドウを継承することもできます．

FIG.2

ディレクトリを作成するためのトランスファ項目

```
Program Title
[ Make Directory         ]
Program Path
[¥COMMAND.COM            ]
Command Line
[$NOSWAP /C MD $PROMPT   ]
```

FIG.3

ディレクトリを削除するためのトランスファ項目

```
Program Title
[Remove Directory        ]
Program Path
[¥COMMAND.COM            ]
Command Line
[$NOSWAP /C RD $PROMPT   ]
```

IDEによるプロジェクトの作成

Windowsから BCX を呼び出すためには，前述の手順で登録した Borlandグループの BCX を選択するだけです.BCX を呼び出したら，File | Change dirでカレントディレクトリをプロジェクトを作成するためのディレクトリに移行します．新しいディレクトリは，DOS Shellで作成してもよいのですが,図2,図3のようなトランスファ項目を設定しておくと便利です．

これらのトランスファ項目を用意すれば，システムメニュー（[GRPH]-[スペース]）から Make Directory を選択し，新しいディレクトリを簡単に作成することができます．もちろん，カレントディレクトリが移動するわけではありませんから，ディレクトリを作成したら，File | Change dir...でそのディレクトリに移動します．しかし，BCX を呼び出すたびにディレクトリを変更しなければならないのは不便ですから，PIF エディタで BCX.PIF ファイルを作成し，初期ディレクトリを設定しておくとよいでしょう．

さて，Windowsアプリケーションを開発するためのプロジェクトを作成しましょう．Project | Open projectで新しいプロジェクト名を作成します．ここでは，HELLOWIN という名前を指定します．これが，最初につくるWindowsアプリケーションです．プロジェクトを作成すると Project ウィンドウがオープンしますので，INS を押してプロジェクトに関連ファイルを追加します．ここでは，HELLOWIN.CPP と WIN.CPP を指定します．各ファイルをオープンするためには，カーソル位置を該当ファイルに

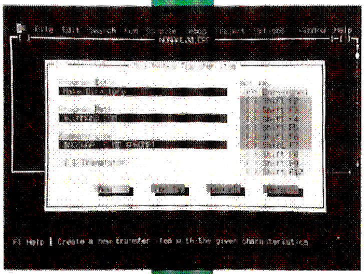


写真3　トランスファ項目の設定

あわせてリターンを押します．

ここで，ヘッダファイルを指定してはいけません．ヘッダファイルは，File | Openでオープンします．プロジェクトを構築すれば，ソースファイルがインクルードしているファイルはProjectウィンドウでスペースを押せばリスト表示されます．

最後に，Options | Applicationダイアログボックスで Windows App を指定します．このダイアログボックスは，Windowsアプリケーションを構築するための各種のオプションを簡易設定するためのものです．ほかのダイアログボックスを使えば，もっと細かな設定を行うこともできますが，小規模のアプリケーションで必要となることはあまりないでしょう．

＊『プログラミング WINDOWS』に掲載されているプログラムをプロジェクトとして指定する場合は，関連する.DEFファイルや.RCファイルもここに指定します．

基本クラスの実装

リスト1（win.h）を見てください．ここでは，C++でWindowsアプリケーションをプログラムするための基本的なクラスが宣言されています．

先頭にある文字列szAppNameの宣言は，このアプリケーション名を表しており，実際にはウィンドウクラスやそのほかのデフォルト名として使用されます．

クラスMainは，Windowsアプリケーション全体に共通する変数や関数

LIST.1

win.h

```
//
//  win.h:
//      Windows アプリケーションのためのクラスを宣言したヘッダファイル
//

#if !defined(__cplusplus)
#error Must specify C++ option.
#endif

#if !defined(__WIN_H)
#define __WIN_H

#if !defined(__WINDOWS_H)
#include <windows.h>
#endif

#if !defined(__STDLIB_H)
#include <stdlib.h>          // for the function 'exit'
#endif

//
//  アプリケーション名の宣言
//      実体をどこかで定義しなければならない。
//
extern char szAppName[];

//
//  この Windows アプリケーション全体に共通する
//  変数や関数のためのクラス
//
class Main {
public:
    static HANDLE hInstance;        // インスタンス
    static HANDLE hPrevInstance;    // 以前に起動したインスタンス
    static LPSTR lpszCmdLine;       // コマンドライン引数
    static int nCmdShow;            // 表示状態
    static int MessageLoop(void);   // メッセージループ
};

//
//  ウィンドウのための基本抽象クラス
//      すべてのウィンドウは、このクラスから派生する。
//      各派生クラスはそれぞれ WndProc を用意しなければならない。
//
class Window {
protected:
    HWND hWnd;      // ウィンドウハンドル
public:
    // ウィンドウハンドルを返す
    HWND GetHandle(void) { return hWnd; }
    // ウィンドウを表示する
    BOOL Show(int nCmdShow) { return ShowWindow(hWnd, nCmdShow); }
    // ウィンドウの内容を更新する
    void Update(void) { UpdateWindow(hWnd); }
    // ウィンドウごとのウィンドウ関数
    virtual long WndProc(unsigned iMessage, WORD wParam, LONG lParam) = 0;
};


//
//  ウィンドウクラスのためのクラス
//      ウィンドウクラスを登録するためには、このアプリケーションに
//      適したウィンドウクラスを指定して Register() を呼び出す
//      派生クラスを用意しなければならない。
//      このクラスは、windows.h の WNDCLASS 構造体から派生している。
//

// このアプリケーションのためのウィンドウ関数のプロトタイプ宣言
long FAR PASCAL _export WndProc(HWND, unsigned, WORD, LONG);

class WinClass : private WNDCLASS {
    BOOL bRegistered;   // ウィンドウクラスの登録状態
public:
    // デフォルトのウィンドウクラスの設定
    WinClass(void) {
        style = CS_HREDRAW | CS_VREDRAW;
        lpfnWndProc = ::WndProc;
        cbClsExtra = 0;
        cbWndExtra = sizeof (Window *);
        hInstance = Main::hInstance;
        hIcon = LoadIcon(NULL, IDI_APPLICATION);
        hCursor = LoadCursor(NULL, IDC_ARROW);
        hbrBackground = GetStockObject(WHITE_BRUSH);
        lpszMenuName = NULL;
        lpszClassName = szAppName;

        bRegistered = FALSE;
    }
    // ウィンドウクラスのパラメータを変更するためのメンバ関数
    void SetClassStyle(WORD newstyle) { style = newstyle; }
    void SetClassWndProc(long (FAR PASCAL *lpfnNewProc)(HWND, unsigned,
            WORD, LONG)) { lpfnWndProc = lpfnNewProc; }
    void SetClassClsExtra(int cbNewClsExtra) { cbClsExtra = cbNewClsExtra; }
    void SetClassWndExtra(int cbNewWndExtra) { cbWndExtra = cbNewWndExtra; }
    void SetClassIcon(HICON hNewIcon) { hIcon = hNewIcon; }
    void SetClassCursor(HCURSOR hNewCursor) { hCursor = hNewCursor; }
    void SetClassBackground(HBRUSH hbrNewBackground)
        { hbrBackground = hbrNewBackground; }
    void SetClassMenuName(LPSTR lpszNewMenu) { lpszMenuName = lpszNewMenu; }
    void SetClassName(LPSTR lpszNewName) { lpszClassName = lpszNewName; }
    LPSTR GetClassName(void) { return lpszClassName; }
    // ウィンドウクラスを登録する
    void Register(void) {
        if (!RegisterClass(this))
            exit(FALSE);
        bRegistered = TRUE;
    }
    // ウィンドウクラスの登録状態を返す
    BOOL isRegistered(void) { return bRegistered; }
};

#pragma warn -aus

#endif  /* __WIN_H */
```

を集めたものです．このクラスのメンバは，すべてstaticで定義されており，単に変数や関数をまとめるためだけにクラスの機能を使っています．

クラスWindowは，さまざまな性質をもつウィンドウのためのクラスを派生する基本クラスとなるものです．

hWndはウィンドウハンドルを示します．頻繁に参照されるメンバなので，派生クラスからアクセスしやすくするためプロテクトメンバとして宣言しています．

最後の仮想関数WndProcは，このクラスから派生したウィンドウが必ず用意しなければならないウィンドウ関数です．ウィンドウ関数は，通常ウィンドウごとに処理内容が違います．性質が異なるものについては，異なるウィンドウ関数をもつクラスを別個に派生しなければなりません．仮想関数として定義されているのは，実際にWindowsから呼び出されるウィンドウ関数は通常1つしかなく，基本クラスから各クラスへのウィンドウ関数を呼び出す分岐処理を行うためです．

WndProcは，_exportとして関数が宣言されています．_exportを記述することで，.DEFファイルのEXPORTS文にWndProcを記述しなくても，エクスポート関数として正しいコードが生成されます．こうしておけば，リンクのための.DEFファイルが必要なくなります．.DEFファイルを使う場合はプロジェクトに.DEFファイル名を指定しますが，指定がなければデフォルトの定義が使われるためです．

ここに用意されたメンバはごく簡単なものでしかありません．ウィンドウの基本要素としてはやや不十分かもしれませんが，この章での説明には十分です．必要なら継承を使って機能を拡張してやることもできるでしょう．

LIST.2

win.cpp

```
#include "win.h"

//
//  Main クラスの本体
//      この Windows アプリケーション全体に共通する変数や関数
//
HANDLE Main::hInstance = 0;         // インスタンス
HANDLE Main::hPrevInstance = 0;     // 以前に起動したインスタンス
LPSTR Main::lpszCmdLine = NULL;     // コマンドライン引数
int Main::nCmdShow = 0;             // 表示状態

//
//  メッセージループ
//
int Main::MessageLoop(void)
{
    MSG msg;

    while (GetMessage(&msg, NULL, 0, 0)) {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }
    return msg.wParam;
}

//
//  Window クラスから派生したクラスのウィンドウ関数を呼び出す
//  ディスパッチ用のウィンドウ関数の定義。
//

//
//  ウィンドウエクストラからクラスへのポインタを取得したり
//  取得するためのインライン関数
//
inline void *GetPointer(HWND hWnd)
{
#if defined(__SMALL__) || defined(__MEDIUM__)
    return (void *)GetWindowWord(hWnd, 0);
#elif defined(__LARGE__) || defined(__COMPACT__)
    return (void *)GetWindowLong(hWnd, 0);
#else
#error use Small, Medium, Compact or Large model
#endif
}

inline void SetPointer(HWND hWnd, void *ptr)
{
#if defined(__SMALL__) || defined(__MEDIUM__)
    SetWindowWord(hWnd, 0, (WORD)ptr);
#elif defined(__LARGE__) || defined(__COMPACT__)
    SetWindowLong(hWnd, 0, (WORD)ptr);
#else
#error use Small, Medium, Compact or Large model
#endif
}

//
//  ウィンドウ関数の本体
//
long FAR PASCAL _export WndProc(HWND hWnd, unsigned iMessage, WORD wParam,
                                LONG lParam)
{
    // ウィンドウエクストラから Window オブジェクトへのポインタを取得する。
    Window *pWin = (Window *)GetPointer(hWnd);

    if (pWin == NULL) {
        if (iMessage == WM_CREATE) {
            LPCREATESTRUCT lpcs;

            // ウィンドウエクストラに Window オブジェクトを
            // 設定する。これによってウインドウエクストラから
            // Window オブジェクトのメンバ関数をアクセスできる。
            lpcs = (LPCREATESTRUCT)lParam;
            pWin = (Window *)lpcs->lpCreateParams;
            SetPointer(hWnd, pWin);
            return pWin->WndProc(iMessage, wParam, lParam);
        }
        return DefWindowProc(hWnd, iMessage, wParam, lParam);
    }
    return pWin->WndProc(iMessage, wParam, lParam);
}
```

クラスWinClassは，windows.hで定義されているWNDCLASSを継承しています．WNDCLASSはウィンドウクラスを登録するための構造体ですが，ウィンドウクラスを1つのオブジェクトとして考えれば，そのためのデータと関数を1つのクラスにまとめることができるわけです．

WNDCLASSは，もともと構造体(struct)として定義されています．読者の皆さんの多くは，C++におけるstructとclassはメンバがデフォルトでパブリック(public，公開メンバ)になるかプライベート(private，非公開

メンバ)になるかが違うだけだという文章をどこかで読まれているでしょう. WNDCLASSをメンバ関数をもたないクラスだと考えれば，ここから新しいクラス(WinClass)を継承することができるのはごくあたりまえのことです．ただし，外部からWinClassのメンバ変数が見えないように，WNDCLASSはprivateとして継承しています．実際には，クラスをプライベートに継承するときはprivateを省略することができます．

コンストラクタWinClassは，クラスWinClassのためのメンバをデフォルト値で初期化しています．実際の，ウィンドウクラスはアプリケーションごとに異なりますからここで直ちに登録はしません．ここから派生したクラスのコンストラクタで，この後ろにあるメンバ関数を呼び出しウィンドウクラスの内容を設定した後，登録するのがよい方法でしょう．あるいは，このクラスのオブジェクトを宣言してメンバ関数Registerを呼び出してもかまいません．

メンバ関数Registerは，ウィンドウクラスを登録します．RegisterClassには自分自身へのポインタthisを渡します．RegisterClassは，windows.hでプロトタイプ宣言がされていますので，自動的にWNDCLASSへのポインタに変換されます．

ここで，メンバ変数cbWndExtraにクラスWindowへのポインタのための領域が確保されている点に注意してください．アプリケーションのウィンドウ関数は，ここで設定されたポインタを受け取ることで，そのポインタが指すウィンドウのオブジェクトを知ることができます．そのオブジェクトのウィンドウ関数を呼び出せば，Windowから派生したどんな性質をもつクラスでも，仮想関数によって正しいウィンドウ関数が呼び出されます．

多くのメンバ関数が，インライン関数として実装されています．インライン関数は，実際にはその場ではコードが生成されず，呼び出されたときにはじめてコードが生成されます．多くの関数は，単独の代入文や別の関数の呼び出しに置き換えられているだけですから，多くのメンバ関数を用意しても，実行ファイルの大きさに影響を与える心配をしなくてすみます．

リスト2 (win.cpp)には，win.hで宣言されているクラスのメンバとウィンドウ関数が定義されています．

クラスMainのメンバは，前述したとおりアプリケーション全体に共通する変数や関数です．

WndProcは，どのクラスのメンバ関数でもなく，このアプリケーションのためのウィンドウ関数です．ウィンドウクラスのメンバcbWndExtraには，クラスWindowへのポインタを格納する領域が確保されており，ウィンドウを作成する際に自分のオブジェクトへのポインタをウィンドウエクストラに格納しておきます．こうしておけば，作成されたウィンドウが，これ以降にメッセージを処理する際は，自分のオブジェクトのためのウィンドウ関数を呼び出して，それぞれに特有の処理をまかせることができるわけです．

WndProcは，ウィンドウエクストラから自分自身へのポインタを取得していますが，このポインタはどこかで正しく設定されていなければ，C++のクラスオブジェクトへのポインタとして使えません．

では，ポインタの設定はどのように行うのでしょうか．ここから派生したクラスがCreateWindowでウィンドウハンドルを作成するとき，最後の引数にオブジェクトへのポインタを指定します（必ず，そうしなければなりません）．WM_CREATEメッセージとともに渡されるlParamは構造体CREATESTRUCTへのポインタで，CreateWindowの最後の引数は，この構造体のlpCreateParamsメンバに渡されます．これを取得して，オブジェクトへのポインタとしてウィンドウエクストラに設定しています．これを行うまでは，WndProcはこのオブジェクトに対するメッセージをすべてDefWndProcへ送ってしまいます．このため，ウィンドウハンドルを作成するまでに送られるメッセージ（WM_GETMINMAXINFO）などは，Windowから派生したクラス固有のウィンドウ関数では処理できません．

なお，ポインタが正しくないことを判定するためにNULLと比較していますが，これは非公開の機能として，ウィンドウエクストラが確保された時点では0にクリアされているためです（Borland C++付属のWINPRIMR.DOCより）．

HelloWin アプリケーション

ウィンドウの基本クラスの実装が終わったら，これを使ってHelloWinアプリケーションを作成します．リスト3にプログラムリストを示します．

まず，文字列szAppNameでHelloWinアプリケーションの名前を定義します．この文字列は，どのアプリケーションでも必ず指定します．

クラスWinAppはウィンドウクラスを登録するためのもので，クラスWinClassを継承しています．HelloWinでは，デフォルトのウインドウクラスをそのまま登録します．

クラスHelloWindowは，Hell-

Windows ソフトウェア一覧
MS-Excel Ver.2.12, 98,000円, スプレッドシート, 日本電気

oWinアプリケーションのためのウィンドウを宣言しています．

コンストラクタHelloWindowは，引数としてウィンドウの中に表示するためのメッセージを指す文字列へのポインタを受け取ります．受け取ったポインタはメンバ変数に代入し，その後でウィンドウを作成し，表示し，内容を更新します．

メンバ関数WndProcは，このクラスのためのウィンドウ関数です．このメンバ関数は，Windowsから直接呼び出されるわけではなく，WinClassで登録したウィンドウ関数::WndProcから呼び出されます．このウィンドウ関数が処理するメッセージは，WM_PAINTとWM_DESTROYだけです．WM_PAINTを受け取るとHelloWindowコンストラクタで設定した文字列をウィンドウに表示します．WM_DESTROYを受け取ると，アプリケーションを終了させるためPostQuitMessageを呼び出します．

WinMainは，DOSプログラムにおけるmain関数に相当する関数で，どのWindowsアプリケーションでも最初に呼び出されます．

ここでは，引数をすべてクラスMainのメンバに代入します．次に，クラスWinAppで宣言したウィンドウクラスを登録するためそのオブジェクトを定義します．さらに，メインウィンドウをオープンするため，クラスHelloWindowのオブジェクトを定義します．"Hello，Windows!"はウィンドウ中に表示する文字列です．最後にメッセージループを呼び出してWinMainから戻ります．

HelloWin アプリケーションの構築

すべてのプログラムが用意できたら，[f•9]（プロジェクトのメイク）を押して，HelloWinアプリケーションを構築します．エラーの修正などについては，マニュアルを参考にしてください．

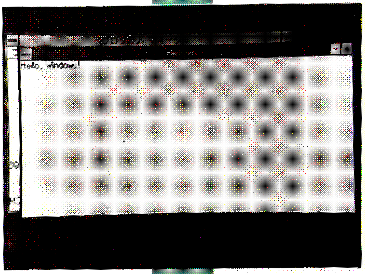


写真4　HelloWin アプリケーションの実行

プログラムの作成に成功しても，IDEからアプリケーションを実行することはできません．[CTRL]-[ESC]を押して，いったんWindowsに戻りプログラムマネージャの「ファイル(F)｜実行(R)」でディレクトリ名も含めてHELLOWIN.EXEを指定して，このアプリケーションを実行してみてください．プログラムが正しく入力されていれば，Hello，Win-

LIST.3

hellowin.cpp

```
//
//  HelloWin アプリケーション
//
#include "win.h"
#include <string.h>

//
//  アプリケーション名の定義
//
char szAppName[] = "HelloWin";

//
//  アプリケーション固有のクラスの定義
//

//
//  HelloWin アプリケーションのためのウィンドウクラスを登録するクラス
//
class WinApp : public WinClass {
public:
    WinApp() { Register(); }
};

//
//  HelloWin アプリケーションが使うウィンドウのためのクラス
//
class HelloWindow : public Window {
    char *szHelloMessage;
public:
    HelloWindow(char *szHello) {
        szHelloMessage = szHello;
        hWnd = CreateWindow(szAppName, szAppName, WS_OVERLAPPEDWINDOW,
            CW_USEDEFAULT, 0,
            CW_USEDEFAULT, 0,
            NULL, NULL, Main::hInstance, (LPSTR)this);
        Show(Main::nCmdShow);
        Update();
    }
    long WndProc(unsigned iMessage, WORD wParam, LONG lParam);
};

//
//  HelloWindow のメインウィンドウのためのウィンドウ関数
//
long HelloWindow::WndProc(unsigned iMessage, WORD wParam, LONG lParam)
{
    switch (iMessage) {
    case WM_PAINT:
        PAINTSTRUCT ps;

        BeginPaint(hWnd, &ps);
        TextOut(ps.hdc, 0, 0, szHelloMessage, strlen(szHelloMessage));
        EndPaint(hWnd, &ps);
        break;

    case WM_DESTROY:
        PostQuitMessage(0);
        break;

    default:
        return DefWindowProc(hWnd, iMessage, wParam, lParam);
    }
    return 0L;
}

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpszCmdLine,
                   int nCmdShow)
{
    // アプリケーション全体に共通する変数の定義
    Main::hInstance = hInstance;
    Main::hPrevInstance = hPrevInstance;
    Main::lpszCmdLine = lpszCmdLine;
    Main::nCmdShow = nCmdShow;

    // アプリケーション固有の部分
    // ウィンドウクラスの登録
    WinApp MyWin;
    // メインウィンドウ
    HelloWindow HelloWin("Hello, Windows!");

    // メッセージループの呼び出し
    return Main::MessageLoop();
}
```

LIST.4

```
// アプリケーション固有の部分
// ウィンドウクラスの登録
WinApp MyWin;
// メインウィンドウ
HelloWindow HelloWin("Hello, Windows!");
HelloWindow HelloWin("Hello, another Windows!");
```

dows!という文字列が表示されたウィンドウが表示されるはずです．うまく動かなかったり，ハングアップしてしまう場合は，プログラムを見なおしてみてください．Windowsアプリケーションのデバッグについては後述します．

プログラムを拡張したり，書き換えたときでも簡単に実行できるよう「ファイル(F)｜新規登録(N)」でHELLOWINをグループに登録しておくと便利です．

前にウィンドウはオブジェクトだといいました．もし，クラスHelloWindowと同じ性質をもつウィンドウをもう1つオープンしたいと思ったら，HelloWindowのオブジェクトをもう1つ定義してやればよいのです．たとえば，リスト3のWinMain関数中でメインウィンドウを定義している箇所を，リスト4のように変更すれば，"Hello, another Windows!"という文字列を表示するウィンドウがもう1つオープンされます．

リソースの作成

Resource Workshopとはなにか

Resource Workshop (RWS) は，視覚的にリソースを編集するためのWindowsアプリケーションです．SDKにもダイアログエディタ，フォントエディタ，ビットマップエディタがありますが，RWSはリソースをプロジェクトとして管理でき，バイナリ・テキストの両方のモードでの編集，アンドゥ，リドゥなど多くの機能をサポートしています．使い勝手は，Borland C++のIDEと同様に使いやすいものです．しかし，残念ながら私はSDKのツールを使い込んでいないので使用感の比較はできません．

RWSは，マニュアルを読まなくてもある程度使うことができ，リソースを編集することができます．おそらく，すべての操作に対するアンドゥ・リドゥ機能だけでも，その使いやすさを感じることができるはずです．

しかし，マニュアルを読まなければ，RWSの本当の実力がわからないのも事実です．メニュー項目の移動や，ダイアログのコントロールの位置合わせなどを簡単に行うため，RWSは実にさまざまな点から，使い勝手を向上させるための手段を用意しています．ここでは，具体的なリソースの作成手順を紹介することで，RWSの実力の一端をご覧いただくことにします．

リソースプロジェクトの作成

RWSも，BCXと同様にプログラムマネージャのBorlandグループに登録しておきます．

RWSを起動したら，できれば最大化ボタンを押してメインウインドウを画面いっぱいにします．とくに，PC-9801のノーマルモードでは解像度が不足しているので，最大化しておかないと編集しにくいと思います．

File｜New Projectで.RCファイルを選択し，新しいリソースプロジェクトを作成します．このプロジェクトには，untitled.rcという名前がつけられます．まず，新しい名前でプロジェクトを保存するためFile｜Save file asで，ghost.rcとしてプロジェクトを保存します．ただし，おそらくカレントディレクトリはプロジェクトとは異なっていますから，プロジェクトとして正しいディレクトリパスを指定します．

ここで，メニューIDなどの識別子を定義するためのヘッダファイルをプロジェクトに追加します．File｜Add file to projectでghost.hを指定すれば，このヘッダファイルがプロジェクトに追加されます．このヘッダファイルはまだ作成していないものなので，「ghost.h does not exit. Create? (ghost.hは存在しません．作成しますか？)」と問い合わせてくるので，Yesを選択します．

次に，このプロジェクトをバイナリリソースとしても保存するため，File｜Preferencesで Multi-saveの.RESチェックボックスをチェックします．リソースファイル名が正しくghost.resとなっているかどうか確認します．

メニューの作成

メニューリソースを作成するためには，Resouce｜NewでResource typeリストボックスからMENUを選択します．メニューエディタのウィ

Windowsソフトウェア一覧
PAL Windows Ver.2, 98,000円, データベース, 大塚商会

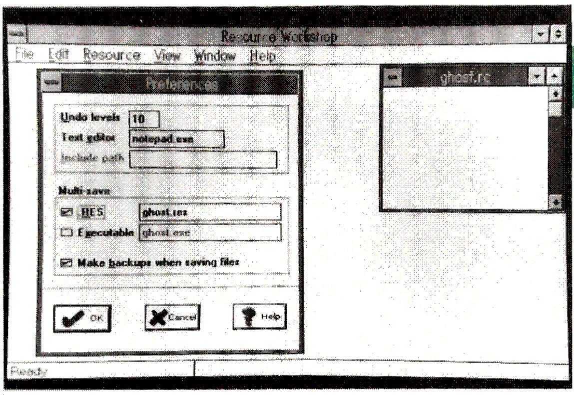


写真 6
File | Preferences ダイアログボックス

ンドウも最大化ボタンを押して，できるだけ広げておきます．

メニューエディタのウィンドウは，左側にあるメニュー項目の詳細を設定するためのダイアログボックスペイン，右上にあるテストペイン，右下にあるメニューの主要な内容だけをリスト表示したり，項目を選択するためのアウトラインペインに分かれます．メニューエディタの中では，どこでも[CTRL]-[↑]，[CTRL]-[↓]を押して項目を移動することができます．

メニュー名は，デフォルトのMENU_1になっているはずです．Resource | Renameを選択して，メニュー名をGHOSTに変更します．「Create a new identifier GHOST?」という問い合わせダイアログではNoを選択します．

カーソルは，ダイアログボックスペインのItem textにあるはずです（なければ，そこをクリックします）．ここにはPop-upというデフォルト名が表示されていますから，これを&Fileと変更します．通常，ポップアップメニューの設定はこれで終わりです．リターンを押して，設定した内容を確定します．

次のメニュー項目に移行するため，[CTRL]-[↓]を押します．マウスでアウトラインペインのMENUITEMをクリックしてもかまいません．このときは，[f･6]を押すかダイアログボックスペインのItem textをクリックします．

ここでは，Item textとして&Newと入力します．このメニュー項目のID値を設定するため，TABを押してItem idに移動しIDM_NEWと入力します．ここで，リターンを押すと「Create a new identifier IDM_NEW?」と表示されますのでYesを選択します．そうすると，New identifierダイアログボックスで値の入力がうながされます．ここでは，101が表示されているはずですので，そのままこの値を採用します．[TAB]を押して，FileをAdd file to projectで設定したghost.hに変更します．設定がすんだら，OKを選択します．

新しいメニュー項目を追加するときは，[INS]を押します．先ほどと同じ手順で，Item textに&Openを，Item idにIDM_OPEN（値=102）を設定します．こんどは，識別子を追加するためにリターンを押すだけでよいはずです．同様にして，&Save（IDM_SAVE，値=103），Save &As(IDM_SAVEAS，値=104)を追加します．

次にメニューセパレータを追加します．現在のSave Asの位置で，[CTRL]-[S]を押します．これで，セパレータが追加されます．

さらに[INS]を押して，A&bout（IDM_ABOUT，値=108）を追加します．New identifierのValue入力ボックスでは108を入力します．ふたたび，[CTRL]-[S]でセパレータを追加し，[INS]を押してE&xit(IDM_EXIT，値=109)を追加します．

これで，Fileメニューができあがりました．右上のテストペインでFileをクリックすれば，今作成したメニューのテストが可能です．ここで項目を選択することで，アウトラインペインのカーソル位置も移動します．

[ESC]を押すか，ほかのペインをクリックしてテストモードから抜け，アウトラインペインの_End Popup_まで移動します．ここで，[CTRL]-[P]を押せば新しいポップアップメニューを追加できます．もし，Exitの項目でポップアップメニューを追加してしまうと，そのポップアップメニューはドロップダウンメニュー項目として追加されてしまいます．

新しく追加したポップアップメニュ

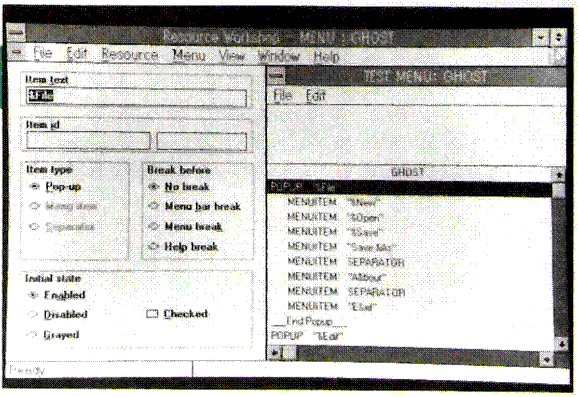


写真 7
メニューリソースの編集

一の項目名を，前述の手順にしたがって&Editとします．メニュー項目のリストとして，&Undo（IDM_UNDO, 値=110），Cu&t（IDM_CUT，値=111），&Copy（IDM_COPY，値=112），&Paste（IDM_PASTE, 値=113），&Delete(IDM_DELETE, 値=114）を追加します．

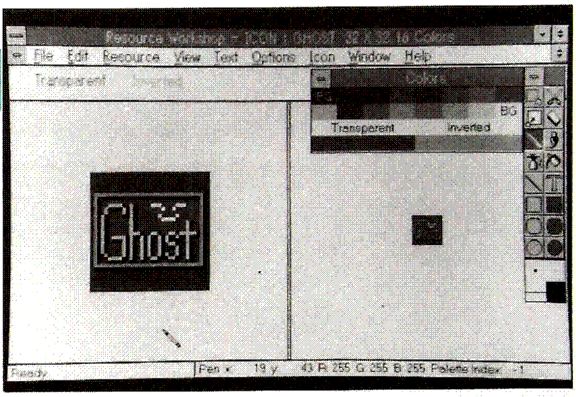
これで，ひととおりメニューができあがりました．File | Save projectを選択してプロジェクトを保存しておきます．せっかくの苦労を水の泡にしないよう，プロジェクトは時々保存しておくようにしたほうがよいでしょう．追加していった識別子の値が重複していないかどうかは，Menu | Check duplicatesで調べることができます．

ここで作成したメニューは，ごく一般的なものです．実は，この一般的なメニュー形式を簡単に指定できるようMenuメニューのメニュー項目にはNew file pop-up，New edit pop-up，New help pop-upという定義済みのポップアップメニューが用意されています．これらは，複数のメニュー項目を含んだポップアップ項目を一度に追加するものです．ただし，空のメニューに追加するときは，一度＿End Menu＿にカーソルを移動してこれらを追加し，その後で先頭のメニューを削除するという手順をとらなければなりません．まったく空のメニューリソースは作成できないためです．識別子の定義はあらためてやりなおさなければなりませんが，先ほどの指定はこれを使えばより簡単にできるでしょう．

すべてうまくいったら，左上のコントロールメニューからClose（クローズ）を選択して，メニューエディタを閉じます．

写真8

アイコンリソースの編集



アイコンの作成

次にアイコンの作成です．Resource | Newを選択し，Resource typeからICONを選択します．

アイコンのようなビットイメージは，ソース（Source）形式かバイナリ（Binary）形式のどちらでリソーススクリプト(.RC)に登録するかを指定しなければなりません．なんらかの理由で作成した.RCファイルを，Microsoftのリソースコンパイラで再度コンパイルしなければならないときはバイナリ（Binary）を選択します．これは，Microsoftのリソースコンパイラが，リソーススクリプト中にソース形式で記述されたビットマップデータをコンパイルできないためです．このときは，バイナリ形式のファイルを格納するためのファイル名の入力が求められます．リソースの再利用は.RESファイルでもできますし，とくに必要がなければ，ソース形式を選択すればよいでしょう．ここでは，ソース形式を選択したものとして話を進めます．

次に，アイコンの大きさと色数を決めるためのダイアログボックスが表示されます．ここでは，Sizeは32×32，Colorsは16 Colorsを選択します（いずれもデフォルト）．

アイコンの形式を設定するとアイコンエディタが起動します．アイコン名がICON_1のようになっていますので，Resource | RenameでGHOSTに変更します．「Create a new identifier GHOST?」という問い合わせではNoを選びます．

アイコンエディタはビットマップエディタの一種ですが，やはり最大化ボタンを押してウィンドウを画面いっぱいにしておくとよいでしょう．また編集画面の拡大表示は，[CTRL]-[Z]を押すかView | Zoom inで倍率をあげられます．

エディタの右側に表示されているアイコンの集まったウィンドウは，ビットマップを編集するためのツールを選択するものです．これらの詳しい説明についてはマニュアルに任せますが，この状態でもアイコンの表示領域をドラッグして描画することができます．編集を誤った場合でも，[GRPH]-[BS]でアンドゥ機能が使えることを忘れないでください．

好みのビットマップを編集できたら，File | Save projectを選択してプロジェクトを保存します．左上のコントロールメニューからClose(クローズ)を選択して，アイコンエディタを閉じ，同様にしてICONウィンドウも閉じます．

Windows ソフトウェア一覧
PLOT-PC WINDOWS 3.0版，100,000円，FORTRANグラフィックライブラリ，三菱総合研究所

ダイアログボックスの作成

次にダイアログボックスを作成します．ここで作成するのは，ダイアログボックスの中でもっとも単純な部類に入るであろうAboutダイアログボックスです．

まず，Resource | NewでResource typeからDIALOGを選択します．次にResource | Renameを使って，DIALOGの名前をGHOSTに変更します．「Create a new identifier GHOST?」という問い合わせではNoを選びます．

ダイアログエディタには，ダイアログボックスを編集するための仮ダイアログボックスが表示されています．

ダイアログボックスのキャプション（仮ダイアログボックスのタイトルバーに表示されているタイトル文字列のこと）は，まだDIALOG_1のようになっているはずです．これを変更するには，仮ダイアログボックスのタイトルバーをクリックして，仮ダイアログボックスを選択状態にします．選択状態になったかどうかは，全体が2重線で囲まれたかどうかでわかります．ここで，About GHOSTと入力すればキャプションが変更されます．

もし，このダイアログボックスをほかのスタイルにしたり使用するフォントを変更したいという場合は，タイトルバーをダブルクリックするかControl | Styleを選んで，表示されるWindow styleダイアログボックスを使います．

先ほど作成したアイコンをこのダイアログボックスに埋め込んでみます．ダイアログエディタの右のほうにToolsという名前のウィンドウがあり，ツールアイコンが並べられています．この中の左から2列目で下から2番目のアイコンを選択します．するとカーソルの形状が変わり，アイコンを配置するモードになります．ここで，仮ダイアログボックスの左上のほうにこれを置きます．アイコンを置いてしまうと，カーソルは通常状態に戻ります．これは，Pick rectangleという矩形で選択範囲を指定するツール（Toolsウィンドウの左上隅）が選択されている状態です．

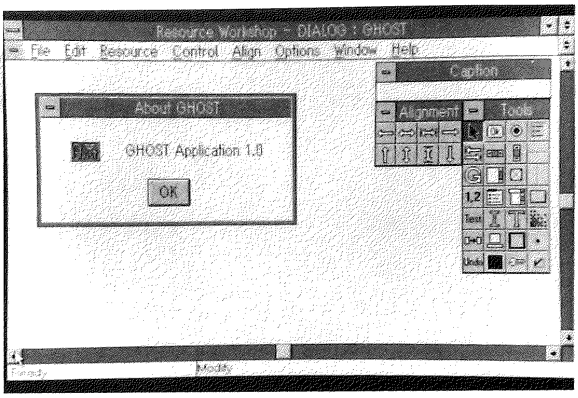


写真9

ダイアログエディタ

アイコンを配置した直後は，このアイコンが選択状態になっています．しかし，いまは四角い灰色が表示されているだけですから，このアイコンのスタイルを指定しなければなりません．アイコンのところでダブルクリックすると，Static styleというダイアログボックスが表示されます．ここでは，Caption入力ボックスにアイコン名GHOSTを入力してOKボタンを押します．これで，アイコンの位置に先ほど作成したGHOSTアイコンが表示されるはずです．

次に文字列を配置してみます．Toolsウィンドウの左から3列目で下から3番目にあるTという形をしたアイコンを選びます．アイコンがTextという形に変わりますので，GHOSTアイコンの右隣にこれを置きます．通常，そのままの状態では文字列を入力するのに短すぎるので，選択枠の右端をドラッグして領域を右方向に伸ばします．ここで，GHOST Application 1.0と入力すれば，実際に表示される文字列が設定できます．もし，すべての文字列を表示するのに領域が不足していたら，選択枠の右端か，反対の左端を横方向にドラッグします．

ダイアログボックス中の配置のために，この文字列を左寄せでなく右寄せにしてみましょう．文字列をダブルクリックすれば，この文字列のためのStatic styleダイアログボックスが表示されます．この中の右側にあるControl typeラジオボタンからRight textをクリックしてOKボタンを押します．

しかし，まだアイコンと文字列とはずれた位置に表示されているはずです．こんどは，これらの位置を揃えてみます．まず，複数の要素をまとめて選択するときは[SHIFT]-クリック（キーボードのSHIFTを押しながら，マウスの左ボタンをクリックする）を使います．マウスだけで範囲を選択するならば，いったん仮ダイアログボックスの外などなにもない位置でクリックして選択を解除してから，仮ダイアログボックスの中の2つの要素をドラッグして枠で囲むようにします．

それぞれの要素を水平位置に並ぶようにしてみましょう．先ほどの要領で2つまとめて選択したら，ダイアログ

エディタの右のほうにあるAlignmentウィンドウから下段の左から2番目にあるアイコンをクリックします．これで枠の中にある要素が垂直方向の中心に揃えられます．次に，上段の右から2番目のアイコンをクリックします．これで，これらの要素がダイアログボックスの水平方向の中心に揃えられました．もし，全体を縦方向だけに移動したいなら，マウスよりも上下のカーソルキーを使う方が安全です．

アイコンと文字列の間が空きすぎていたり狭すぎる場合は，[CTRL]を押しながら選択枠の右端を横方向にドラッグします．これらは自動的に移動します．この後で，先ほどのAlignmentウィンドウのアイコンをクリックすれば，また中央に揃えることができます．

最後にボタンを追加します．Toolsウィンドウの左から2番目のいちばん上にあるOKの形をしたアイコンをクリックします．仮ダイアログボックスに配置されたアイコンと文字列の下に，このボタンを置きます．このボタンをダブルクリックして，Button styleダイアログボックスを表示させます．ここで，CaptionとしてOKを入力します．[TAB]を2回押してControl idに移動しIDOKを入力します．さらにTABを7回押してButton typeラジオボタンをPush buttonからDefault push buttonに変更します．設定がすんだらOKボタンを押します．

ボタンを水平方向の中央に揃えるため，Alignmentウィンドウの上段の右から2番目のアイコンをクリックします．

おそらく，ダイアログボックスは縦方向に長すぎるでしょう．仮ダイアログボックスのタイトルバーをクリックしてこれを選択状態にし，下辺をドラッグして適当な大きさに変更します．

これで，ダイアログボックスの設計は終わりました．File | Save projectでいったんリソースプロジェクトを保存します．ダイアログボックスのテストを行いたい場合は，Toolsウィンドウの左列の下から3番目にあるTestアイコンをクリックします．リターンを押せばテストモードから抜けられます．

作成されたリソースファイル

正しくオプションが設定されていれば，リソースに関してghost.rcのほかにRWSのプロジェクトファイルを表すghost.rws，バイナリリソースファイルghost.res，ヘッダファイルghost.hができあがっているはずです．

ghost.rcとghost.hは，それぞれリスト5，6に示されるテキストファイルになります．ghost.rcの中ほどにあるビットイメージ情報は，Microsoftのリソースコンパイラではコンパイルできない形式のものです．ghost.hは，次に作成するghost.cppのプログラムからも参照されます．なお，このヘッダファイルにコメントをいれても無意味です．次に，RWSを起動して識別子を追加・修正したときにそれらのコメントは削除されてしまいます．

リソースを使ったWindowsアプリケーション

GHOSTアプリケーション

前節で作成したリソースを使ってGHOSTアプリケーションを作成します．

まず，BCXを呼び出してProject | Open projectを選択し，新しいプロジェクトとしてGHOSTと入力します．ProjectウィンドウにはGHOST.CPPとWIN.CPP，そして前節で作成したGHOST.RESを指定します．さらに，Options | ApplicationでWindows Appを指定します．GHOST.CPPのリストをリスト7に示します．

ここでは，まずモードつきダイアログボックスのための基本クラスModalDialogを宣言しています（本来，ここからアプリケーション名の定義の前までのモードつきダイアログボックスに関するクラスについては，win.hに入れておくべきものですが，解説の都合上この場所で定義しています）．

ダイアログボックスは，通常ウィン

LIST.5

ghost.h

```
#define IDM_NEW 101
#define IDM_OPEN     102
#define IDM_SAVE     103
#define IDM_SAVEAS   104
#define IDM_ABOUT    108
#define IDM_EXIT     109
#define IDM_UNDO     110
#define IDM_CUT 111
#define IDM_COPY     112
#define IDM_PASTE    113
#define IDM_DELETE   114
```


Windows ソフトウェア一覧
Qmail, 148,000円, 電子メールシステム, ミナトエレクトロニクス

LIST.6

ghost.rc

```
#include "ghost.h"
GHOST MENU
BEGIN
    POPUP "&File"
    BEGIN
        MENUITEM "&New", IDM_NEW
        MENUITEM "&Open", IDM_OPEN
        MENUITEM "&Save", IDM_SAVE
        MENUITEM "Save &As", IDM_SAVEAS
        MENUITEM SEPARATOR
        MENUITEM "A&bout", IDM_ABOUT
        MENUITEM SEPARATOR
        MENUITEM "E&xit", IDM_EXIT
    END

    POPUP "&Edit"
    BEGIN
        MENUITEM "&Undo", IDM_UNDO
        MENUITEM SEPARATOR
        MENUITEM "Cu&t", IDM_CUT
        MENUITEM "&Copy", IDM_COPY
        MENUITEM "&Paste", IDM_PASTE
        MENUITEM "&Delete", IDM_DELETE
    END

END
GHOST ICON
BEGIN
    '00 00 01 00 01 00 20 20 10 00 00 00 00 00 E8 02'
    '00 00 16 00 00 00 28 00 00 00 20 00 00 00 40 00'
    '00 00 01 00 04 00 00 00 00 00 80 02 00 00 00 00'
    '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
    '00 00 00 00 80 00 00 80 00 00 00 80 80 00 80 00'
    '00 00 80 00 80 00 80 80 00 00 80 80 80 00 C0 C0'
    'C0 00 00 00 FF 00 00 FF 00 00 00 FF FF 00 FF 00'
    '00 00 FF 00 FF 00 FF FF 00 00 FF FF FF 00 00 00'
    '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
    '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
    '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
    '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
    '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
    '00 00 00 00 00 00 00 00 00 00 00 00 00 00 BB BB'
    'BB BB BB BB BB BB BB BB BB BB BB BB BB B0 B0 33'
    '00 00 03 00 30 03 00 03 30 00 33 00 33 B0 B0 30'
    'EE EE 00 E0 3E 03 EE 00 0E E0 03 E0 33 B0 B0 0E'
    '03 33 E0 E0 3E 0E 03 E0 E3 3E 03 E0 33 B0 B0 E0'
    '33 33 E0 E0 3E 0E 03 E0 33 3E 03 E0 33 B0 B0 E0'
    '33 33 E0 E0 3E 0E 03 E0 30 0E 03 E0 33 B0 B0 E0'
    '33 33 E0 E0 3E 0E 03 E0 3E E0 33 E0 33 B0 B0 E0'
    '30 00 E0 E0 3E 0E 03 E0 E0 30 03 E0 33 B0 B0 E0'
    '3E EE E0 E0 0E 0E 00 E0 E0 0E 00 E0 00 B0 B0 E0'
    '33 33 33 EE E0 33 EE 03 3E E0 EE EE E0 B0 B0 E0'
    '33 33 33 E0 33 33 33 33 33 33 33 E0 33 B0 B0 E0'
    '33 33 33 E0 33 33 33 33 33 33 33 E0 33 B0 B0 E0'
    '33 33 33 E0 33 33 33 33 33 33 33 E0 33 B0 B0 E0'
    '33 33 33 E0 33 33 33 3F FF F3 33 33 33 B0 B0 E0'
    '33 33 00 E0 33 33 33 F3 33 3F 33 33 33 B0 B0 3E'
    '03 33 E0 E0 33 33 33 33 33 33 33 33 33 B0 B0 3E'
    '00 00 E0 E0 33 33 33 3F 33 F3 33 33 33 B0 B0 33'
    'EE EE 03 E0 33 33 FF F3 33 3F FF 33 33 B0 B0 33'
    '33 33 33 33 33 33 33 33 33 33 33 33 33 B0 B0 00'
    '00 00 00 00 00 00 00 00 00 00 00 00 00 B0 BB BB'
    'BB BB BB BB BB BB BB BB BB BB BB BB BB B0 00 00'
    '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
    '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
    '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
    '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
    '00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF FF'
    'FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF'
    'FF FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
    '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
    '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
    '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
    '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
    '00 00 00 00 00 00 00 00 00 00 FF FF FF FF FF FF'
    'FF FF FF FF FF FF FF FF FF FF FF FF FF FF'
END
GHOST DIALOG 19, 28, 142, 51
CAPTION "About GHOST"
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
BEGIN
    CONTROL "GHOST", -1, "STATIC", SS_ICON, 17, 9, 16, 14
    CONTROL "GHOST Application 1.0", -1, "STATIC", SS_RIGHT | WS_GROUP, 39, 12, 86, 8
    CONTROL "OK", IDOK, "BUTTON", BS_PUSHBUTTON | WS_TABSTOP, 59, 30, 24, 14
END
```

ドウとは異なりダイアログボックスはウィンドウエクストラを指定できないので，派生したダイアログボックスクラスのオブジェクトに適当なダイアログボックス関数を呼び出すよう内部に静的ポインタを用意しています．

クラスModalDialogのコンストラクタは，有効なダイアログボックスオブジェクトへのポインタを静的メンバcurDlgに代入し，引数として受け取った親ウィンドウのハンドルをメンバ変数に代入して，ダイアログボックスの準備をします．実際に，ダイアログボックスの処理を実行するのはメンバ関数runです．本来のダイアログボックス関数DispatchDlgProcはコールバック関数と呼ばれ，Windowsから呼び出されます．この関数も_exportで宣言しなければなりません．ここから，クラスModalDialogから派生した各クラスのダイアログボックス関数を呼び出します．

クラスModalDialogの実装には，若干問題があります．アプリケーションの実行中ある時点で有効なダイアログボックスは，静的メンバcurDlgが指す1つだけしか認められないためです．モードつきダイアログボックスですから，通常はダイアログボックスから抜けるまでは，ほかのダイアログボックスは表示されないはずですが，複数のウィンドウからダイアログボックスを呼び出した場合などで，2つのダイアログボックスを切り換えたりするような場合には対応できません．

アプリケーション名は，"GHOST"とします．

GHOSTアプリケーションのウインドウクラスには，メニューとしてアプリケーション名と同じ"GHOST"を指定します．

クラスAboutDialogは，Aboutダイアログボックスのためのものです．このクラスは，ModalDialogから派生しています．ダイアログ関数は，コマンドとしてIDOKが返された場合に，ダイアログボックスを終了させることだけを処理します．前述のように，ダイアログボックスはオブジェクトを定義しただけでは実行されません．宣言したオブジェクトのrunメンバ関数を呼び出します．実行を終えたらオブジェクトは削除します．ここでは，オブジェクトの構築と削除にnewとdeleteを使っています．

GHOSTアプリケーションのメインウィンドウのクラスは，HelloWinアプリケーションと同じです．異なるのはウィンドウ関数で，メニューを選択することで返されるWM_COMMANDメッセージを処理しています．ただし，実際に処理されるのはFile | AboutとFile | Exitだけです．その

LIST.7

ghost.cpp

```
#include "win.h"
#include "ghost.h"

//
//  モードつきダイアログのためのクラス
//      モードつきダイアログボックスは、このクラスから派生する。
//      各派生クラスはそれぞれ DlgProc と GetDialogName を
//      用意しなければならない。
//
class ModalDialog {
    static ModalDialog *curDlg;     // 現在オープン中のダイアログボックスを指す
    ModalDialog *preDlg;            // その前にオープンされていたダイアログ
    HWND hParentWnd;                // ダイアログボックスの親ウィンドウハンドル
protected:
    HWND hDlg;                      // ダイアログボックスのためのハンドル
public:
    ModalDialog(HWND hWnd) {        // ダイアログボックスの初期設定
        preDlg = curDlg;
        curDlg = this;
        hParentWnd = hWnd;
    }
    ~ModalDialog() {                // ダイアログボックスの終了
        curDlg = preDlg;
    }
    void run(void);                 // ダイアログボックスの実行
    // ふさわしいダイアログボックス関数を呼び出すためのダイアログボックス関数
    static BOOL FAR PASCAL _export DispatchDlgProc(HWND hCurDlg,
                unsigned iMessage, WORD wParam, LONG lParam);
    // ダイアログボックスごとのダイアログボックス関数
    virtual BOOL DlgProc(unsigned iMessage, WORD wParam, LONG lParam) = 0;
    // 派生したクラスが返差なければならないダイアログボックス名
    virtual LPSTR GetDialogName(void) = 0;
};

//
//      モードつきダイアログボックスのクラスの本体
//
ModalDialog *ModalDialog::curDlg = NULL;

// ダイアログボックスの実行
void ModalDialog::run(void) {
    FARPROC lpfnDlgProc;

    // インスタンスサンクの作成
    lpfnDlgProc = MakeProcInstance((FARPROC)ModalDialog::DispatchDlgProc,
                            Main::hInstance);
    // ダイアログボックス関数の呼び出し
    DialogBox(Main::hInstance, GetDialogName(), hParentWnd, lpfnDlgProc);
    // インスタンスサンクの解放
    FreeProcInstance(lpfnDlgProc);
}

// Windowsからコールバックされるダイアログボックス関数
BOOL FAR PASCAL _export ModalDialog::DispatchDlgProc(HWND hCurDlg,
                unsigned iMessage, WORD wParam, LONG lParam)
{
    curDlg->hDlg = hCurDlg;
    return curDlg->DlgProc(iMessage, wParam, lParam);
}

//
//  アプリケーション名の定義
//
char szAppName[] = "GHOST";

//
//  アプリケーション固有のクラスの定義
//

//
//  Ghost アプリケーションのウィンドウクラスを登録するクラス
//
class WinApp : public WinClass {
public:
    WinApp() {
        SetClassIcon(LoadIcon(Main::hInstance, szAppName));
        SetClassMenuName(szAppName);    // メニューの設定
        Register();
    }
};

//
//  About ダイアログボックス
//
class AboutDialog : public ModalDialog {
public:
    AboutDialog(HWND hWnd) : ModalDialog(hWnd) {}
    BOOL DlgProc(unsigned iMessage, WORD wParam, LONG lParam);
    LPSTR GetDialogName(void) { return szAppName; }
};

//
//  About ダイアログボックスのためのダイアログ関数
//
#pragma argsused

BOOL AboutDialog::DlgProc(unsigned iMessage, WORD wParam, LONG lParam)
{
    switch (iMessage) {
    case WM_INITDIALOG:
        break;

    case WM_COMMAND:
        switch (wParam) {
        case IDOK:
            EndDialog(hDlg, 0);
            break;
        default:
            return FALSE;
        }
        break;

    default:
        return FALSE;
    }
    return TRUE;
}

//
//  Ghost アプリケーションが使うウィンドウのためのクラス
//
class GhostWindow : public Window {
public:
    GhostWindow() {
        hWnd = CreateWindow(szAppName, szAppName, WS_OVERLAPPEDWINDOW,
            CW_USEDEFAULT, 0,
            CW_USEDEFAULT, 0,
            NULL, NULL, Main::hInstance, (LPSTR)this);
        Show(Main::nCmdShow);
        Update();
    }
    long WndProc(unsigned iMessage, WORD wParam, LONG lParam);
};

//
//  Ghost のメインウィンドウのためのウィンドウ関数
//
long GhostWindow::WndProc(unsigned iMessage, WORD wParam, LONG lParam)
{
    switch (iMessage) {
    case WM_PAINT:
        static char szHello[] = "Hello, Windows!";
        PAINTSTRUCT ps;

        BeginPaint(hWnd, &ps);
        TextOut(ps.hdc, 0, 0, szHello, sizeof szHello - 1);
        EndPaint(hWnd, &ps);
        break;

    case WM_COMMAND:
        switch (wParam) {
        case IDM_ABOUT:
            // About ダイアログボックスの表示
            AboutDialog *pAboutDlg = new AboutDialog(hWnd);
            pAboutDlg->run();
            delete pAboutDlg;
            break;

        case IDM_NEW:
        case IDM_OPEN:
        case IDM_SAVE:
        case IDM_SAVEAS:
        case IDM_UNDO:
        case IDM_CUT:
        case IDM_COPY:
        case IDM_PASTE:
        case IDM_DELETE:
            // これらの機能は実現されていない
            MessageBox(hWnd, "This feature is not available.",
                szAppName, MB_ICONASTERISK | MB_OK);
            break;

        case IDM_EXIT:
            // GHOST アプリケーションを終了する
            SendMessage(hWnd, WM_CLOSE, 0, 0L);
            break;
        }
        break;

    case WM_DESTROY:
        PostQuitMessage(0);
        break;

    default:
        return DefWindowProc(hWnd, iMessage, wParam, lParam);
    }
    return 0L;
}

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpszCmdLine,
                int nCmdShow)
{
    // アプリケーション全体に共通する変数の定義
    Main::hInstance = hInstance;
    Main::hPrevInstance = hPrevInstance;
    Main::lpszCmdLine = lpszCmdLine;
    Main::nCmdShow = nCmdShow;

    // アプリケーション固有の部分
    // ウィンドウクラスの登録
    WinApp MyWin;
    // メインウィンドウ
    GhostWindow GhostWin;

    // メッセージループの呼び出し
    return Main::MessageLoop();
}
```


Windows ソフトウェア一覧
SQL Windows 2.0 (英語版), 298,000円, データベースおよびプログラム開発ツール, 三菱商事

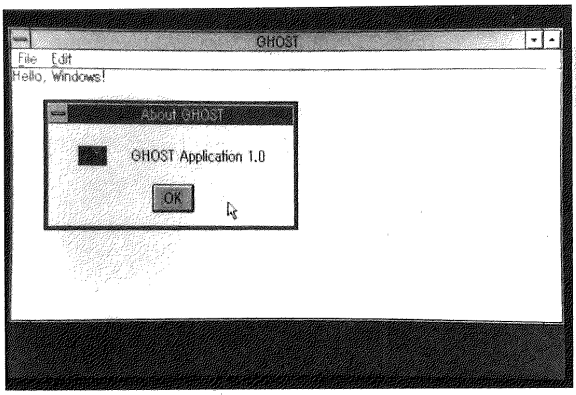


写真10
GHOSTアプリケーション

ほかのメニュー項目は，選択してもメッセージボックスが表示されるだけで何の機能も実現されていません（GHOSTというアプリケーション名は，ここからきています）．

WinMainは，HelloWinアプリケーションと同じ形のものです．

クリップボードを使ったアプリケーション

SqueezeWinアプリケーション

DDEに関する処理は，アプリケーション間のメッセージの交換が中心になります．これまで，メッセージ処理はクラス固有のウィンドウ関数に任せていましたから，DDEのためのC++的な応用法はあまりないようにみえます．実は，Borland C++はC++の仮想関数を拡張して，仮想関数が呼び出されるときのWindowsのメッセージを指定できるようになっているそうなのですが，いまのところObject Windows Libraryがその機能を使っているらしいということしかわかりません（OOP TOURの資料より）．

そのかわり，クリップボードを扱うアプリケーションとして，ウィンドウ全体を圧縮して自分のウィンドウに表示するSqueezeWinアプリケーションを作成します．DDEについても，これと同じように手続きをまとめてクラス化したり，それを継承して機能を拡張するということができるでしょう．

これまでと同じように，BCXを呼び出してProject | Open projectを選択し，新しいプロジェクトとしてSQUEEZEWと入力します．Projectウィンドウには，SQUEEZEW.CPPとWIN.CPPを指定します．Options | ApplicationでWindows Appを指定することを忘れないでください．SQUEEZEW.CPPのリストをリスト8に示します．

ここでは，クリップボードに関する処理をクラスClipboardにまとめています．このクラスには，テキスト，ビットマップ，メタファイルピクチャをクリップボードとやりとりするための手続きが用意されています．これ以外の処理が必要な場合は，やはりこのクラスを継承します．クリップボードに対する基本操作がプロテクトメンバ関数になっているのはそのためです．クリップボードに対する操作は，このクラスのオブジェクトを宣言してそれに対して行うことにします．

アプリケーション名は"SqueezeWin"とします．

SqueezeWinアプリケーションの構成は，基本的にはウィンドウ関数以外はHelloWinと同じものです．

ウィンドウ関数は，マウスが押されたときに画面全体のイメージをビットマップとしてクリップボードに転送する処理と，描画が要求されたときにクリップボードからビットマップを取り出し自分のウィンドウに処理します．

このウィンドウの中でマウスの左ボタンが押されると，ウィンドウ関数にWM_LBUTTONDOWNメッセージが渡されます．ここでは，GetDC（NULL）で画面全体のデバイスコンテキストを取得します（これは非公開の機能として『プログラミングWINDOWS』に記載されています）．また，メモリデバイスコンテキストと，画面全体の大きさと同じビットマップを作成します．ビットマップの作成に成功すると，画面全体のイメージのビットマップを転送します．ビットマップを，クリップボードに渡すにはクラスClipboardのオブジェクトを作成して，ビットマップハンドルをメンバSetBitmapに渡すだけです．このオブジェクトはこのブロックを抜けるとなくなってしまいますが，クリップボードはWindows自身が管理するものですのでなくなるわけではありません．ビットマップハンドルはクリップボードが管理します．もし，ビットマップが作成できなければビープ音を発してエラーを示します．

ウィンドウを描画するためにWM_PAINTメッセージが渡されると，クリップボードからビットマップを取り出します．ビットマップをウィンドウ中に描画するために，StretchBltを使って現在のウィンドウの大きさに圧縮しています．ウィンドウの大きさは，

LIST.8

squeezew.cpp

```
//
//  SqueezeWin アプリケーション
//
#include "win.h"

//
//  クリップボードのためのクラス
//
class Clipboard {
    int hWnd;   // クリップボードを使用するウィンドウのハンドル
    BOOL bOpen; // クリップボードがオープンかどうかを示すフラグ
protected:
    // クリップボードにハンドルが示すデータを設定する
    void SetData(WORD wFormat, HANDLE hMem) {
        OpenClipboard(hWnd);
        EmptyClipboard();
        SetClipboardData(wFormat, hMem);
        CloseClipboard();
    }
    // クリップボードからデータハンドルを取得する
    // データハンドルはクリップボードが管理し、
    // オープン中は他のプログラムに制御を渡してはいけない。
    HANDLE GetData(WORD wFormat) {
        HANDLE hMem;

        OpenClipboard(hWnd);
        hMem = GetClipboardData(wFormat);
        bOpen = TRUE;
        return hMem;
    }
public:
    Clipboard(HWND hCurWnd) {
        hWnd = hCurWnd;
        bOpen = FALSE;
    }
    // クリップボードをクローズする
    //      クリップボードからデータを取り出したら、他のプログラムに
    //      制御を渡す前に、クリップボードを閉じなければならない。
    void Close(void) {
        if (bOpen) {
            CloseClipboard();
            bOpen = FALSE;
        }
    }
    ~Clipboard() { Close(); }
    // クリップボードと各種のデータをやりとりする関数
    void SetText(HANDLE hMem) { SetData(CF_TEXT, hMem); }
    HANDLE GetText(void) { return GetData(CF_TEXT); }
    void SetBitmap(HBITMAP hBitmap) { SetData(CF_BITMAP, hBitmap); }
    HBITMAP GetBitmap(void) { return GetData(CF_BITMAP); }
    void SetMetafile(HANDLE hMem) { SetData(CF_METAFILEPICT, hMem); }
    HANDLE GetMetafile(void) { return GetData(CF_METAFILEPICT); }
};

//
//  アプリケーション名の定義
//
char szAppName[] = "SqueezeWin";

//
//  アプリケーション固有のクラスの定義
//

//
//  SqueezeWin アプリケーションのためのウィンドウクラスを登録するクラス
//
class WinApp : public WinClass {
public:
    WinApp() { Register(); }
};

//
//  SqueezeWin アプリケーションが使うウィンドウのためのクラス
//
class SqueezeWindow : public Window {
public:
    SqueezeWindow() {
        hWnd = CreateWindow(szAppName, szAppName, WS_OVERLAPPEDWINDOW,
            CW_USEDEFAULT, 0,
            CW_USEDEFAULT, 0,
            NULL, NULL, Main::hInstance, (LPSTR)this);
        Show(Main::nCmdShow);
        Update();
    }
    long WndProc(unsigned iMessage, WORD wParam, LONG lParam);
};

//
//  SqueezeWin のメインウィンドウのためのウィンドウ関数
//
long SqueezeWindow::WndProc(unsigned iMessage, WORD wParam, LONG lParam)
{
    static short xClient, yClient;
    HDC hDC, hMemDC;
    BITMAP bm;
    HBITMAP hBitmap;

    switch (iMessage) {
    case WM_SIZE:
        // ウィンドウの大きさを取得する
        xClient = LOWORD(lParam);
        yClient = HIWORD(lParam);
        break;

    case WM_LBUTTONDOWN:
        // マウスの左ボタンが押されたときの処理

        // ウィンドウ全体のデバイスコンテキストを取得する
        hDC = GetDC(NULL);
        // ウィンドウ全体のビットイメージをクリップボードに設定する
        hMemDC = CreateCompatibleDC(hDC);
        hBitmap = CreateCompatibleBitmap(hDC,
            GetSystemMetrics(SM_CXSCREEN),
            GetSystemMetrics(SM_CYSCREEN));
        if (hBitmap) {
            SelectObject(hMemDC, hBitmap);
            BitBlt(hMemDC, 0, 0, GetSystemMetrics(SM_CXSCREEN),
                GetSystemMetrics(SM_CYSCREEN),
                hDC, 0, 0, SRCCOPY);

            // 使用するクリップボードの作成
            Clipboard Clip(hWnd);
            // ビットマップの設定
            Clip.SetBitmap(hBitmap);
            InvalidateRect(hWnd, NULL, TRUE);
        } else {
            MessageBeep(0);
        }
        DeleteDC(hMemDC);
        ReleaseDC(hWnd, hDC);
        break;

    case WM_PAINT:
        PAINTSTRUCT ps;
        Clipboard Clip(hWnd);

        hDC = BeginPaint(hWnd, &ps);
        // クリップボードからビットマップを取得
        hBitmap = Clip.GetBitmap();
        if (hBitmap) {
            // ビットマップをウィンドウ中に描画
            SetCursor(LoadCursor(NULL, IDC_WAIT));
            hMemDC = CreateCompatibleDC(hDC);
            SelectObject(hMemDC, hBitmap);
            GetObject(hBitmap, sizeof (BITMAP), (LPSTR)&bm);
            StretchBlt(hDC, 0, 0, xClient, yClient,
                hMemDC, 0, 0, bm.bmWidth, bm.bmHeight, SRCCOPY);
            SetCursor(LoadCursor(NULL, IDC_ARROW));
            DeleteDC(hMemDC);
        }
        // クリップボードのデータの使用を終了
        Clip.Close();
        EndPaint(hWnd, &ps);
        break;

    case WM_DESTROY:
        PostQuitMessage(0);
        break;

    default:
        return DefWindowProc(hWnd, iMessage, wParam, lParam);
    }
    return 0L;
}

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpszCmdLine,
                   int nCmdShow)
{
    // アプリケーション全体に共通する変数の定義
    Main::hInstance = hInstance;
    Main::hPrevInstance = hPrevInstance;
    Main::lpszCmdLine = lpszCmdLine;
    Main::nCmdShow = nCmdShow;

    // アプリケーション固有の部分
    // ウィンドウクラスの登録
    WinApp MyWin;
    // メインウィンドウ
    SqueezeWindow SqueezeWin;

    // メッセージループの呼び出し
    return Main::MessageLoop();
}
```

WM_SIZEメッセージを処理して取得しています．クリップボードから取り出したデータを使い終わったらメンバCloseを呼び出します．

一般に，クリップボードからデータを取り出しても，そのデータの管理はやはりクリップボードが行います．Clipboardクラスのオブジェクトからデータを取り出したら，Closeを呼

Windows ソフトウェア一覧
TBゴシックB，33,000円，太ゴシック体アウトラインフォント，アルプス電気

び出すまではほかのプログラムに制御を切り換えてはいけません．つまり，メッセージ間でクリップボードのデータを処理したり，SendMessageやPeekMessageを呼び出してはいけないということです．こうした処理が必要ならば，取り出したデータのコピーを作成しなければなりません．

WM_PAINTメッセージを処理するときに使われるクラスClipboardのオブジェクトClipは，ウィンドウ関数が終了するまで有効です．Clipが有効な領域をWM_PAINTメッセージの間だけに限定したい場合は，このメッセージの処理部分をブロック({})で囲んでやるか，Clipboard * pClip = new Clipboard(hWnd);としてメッセージ処理の最後でdelete pClip;としてやるとよいでしょう．このとき，メンバの呼び出しはpClip->GetBitmap();のようにしなければなりません．

まとめ

C++によるWindowsアプリケーションの開発は，まさにウィンドウの性質をクラスのメンバに反映させることです．基本的な性質は基本クラスにまとめ，そこから継承を使って各ウィンドウ固有の機能を自然に拡張することができます．ここでは，複数の種類のウィンドウをもつアプリケーションを示すことができませんでしたが，実際のアプリケーションの多くは異なるウィンドウをいくつももっていますし，別のWindowsアプリケーションを開発する場合にも既存のクラスをそのまま応用して自分のクラスとして使うことができます．

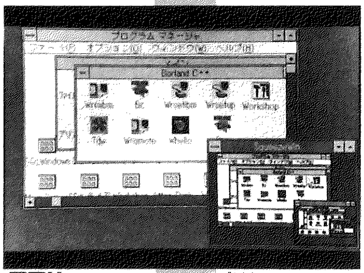


写真11　SqueezeWin アプリケーション

また，クラスは目に見えるものだけに利用されるものではありません．win.hでは，アプリケーション全体に共通する変数や手続きをまとめるためにMainクラスを定義していますし，SqueezeWinアプリケーションでは，クリップボードの手続きをクラス化しました．これらをクラス化することで，現在行っている処理の目的をほかの処理と明確に分離できます．

C++は，オブジェクト指向以外にも数多く改良された点があります．これらの多くが，アプリケーション開発に役立つことでしょう．

デバッグについて

Turbo Debugger for Windows

Borland C++にバンドルされているTurbo Debugger for Windows(TDW)は，まぎれもなくWindowsアプリケーションです．しかし，同時に特殊なWindowsアプリケーションであることも確かで，TDWはほかのWindowsアプリケーションとは異なりCUI (Character User Interface)環境です．もっとも，SDKに付属しているCodeView for Windowsも同様です．

TDWのメリットは，DOS版のTurbo Debugger(TD)と同様のユーザーインターフェースがそのまま受け継がれていることです．TDWでは，さらにWindows用に特定の種類あるいはすべてのウィンドウメッセージを記録する機能が追加されています．

Borlandグループへの登録

TDWを，BCXと同じようにBorlandグループに登録します．このとき，TDWのコマンド名の後ろに-tオプションを指定することができます．このオプションは，TDWが呼び出されたときの初期ディレクトリを指定するためのもので，これを使えばPIFファイルで初期ディレクトリを指定する必要はありません．

TDWの詳しい使い方はマニュアルに任せますが，基本的な操作はDOS版のTDと同じです．なお，TDWをカスタマイズするためには，DOSコマンドラインでTDINST -wと入力します．

GHOSTアプリケーションのデバッグ

TDWで追加された機能でもっとも目をひくのは，View | Window messagesウィンドウでしょう．このウィンドウを使って，ウィンドウハンドルかウィンドウ関数とメッセージの種類を指定し，そのウィンドウに送られたメッセージを記録したり，メッセージが送られた時点でブレーク(停止)することができます．

先ほど作成したGHOSTアプリケーションをTDWでデバッグしてみましょう．オプションを変更していなければ実行ファイルにデバッグ情報が

Windows ソフトウェア一覧
TB明朝M，33,000円，中明朝体アウトラインフォント，アルプス電気

付加されているはずです．デバッグオプションをオフにしてプロジェクトを構築した場合は，オンにして再構築してください．

プログラムマネージャのBorlandグループから，TDWを選択して呼び出します．TDWが起動しても画面にはなにも表示されません．File | Openでデバッグの対象プログラムを指定するまでは，CPUウィンドウさえもオープンできません．View | Window messagesを選択し，Window messagesウィンドウをオープンします．左上のペインでWndProcと入力すれば，このアプリケーションのウィンドウ関数を指示できます．

TABを押して，下のペインに移動します．ここには指定されたメッセージが記録されます．この内容をログファイルに記録するためには，ここで[CTRL]-[S]を押してローカルメニューのSend to log windowをYesに変更します（[CTRL]-[f･10]を押せば確認できます）．次に，View | LogでLogウィンドウを表示し，[CTRL]-[O]を押してログファイルをオープンします．これで，受け取ったメッセージが随時ここでオープンしたログファイルに記録されます．

Window messagesウィンドウに戻り，[TAB]を押して右上のペインに移動します．ここには，デフォルトでLog all messagesと表示されています．しかし，すべてのメッセージを取得するとかなり負担が重くなってしまいます．ここでは，WM_COMMANDと入力してこのメッセージだけを記録するようにします．このメッセージはメニューで選択された項目のIDをウィンドウ関数のwParam引数に渡すものです．

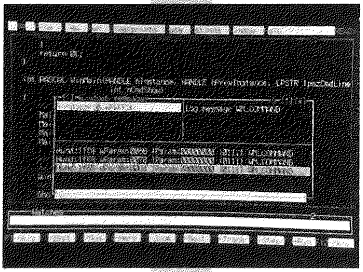


写真12　Turbo Debugger for Windows

準備ができたら[f･9]を押してGHOSTアプリケーションを実行します．ウィンドウメッセージを監視しているときは，通常の実行よりも遅くなってしまいますが，これはしかたがありません．GHOSTアプリケーションのメニューから，いくつかのメニューを選択してみてください（ほとんどのメニュー項目は，機能が実現されていないというメッセージボックスが表示されるだけです）．

GHOSTアプリケーションを終了したら，Window messagesウィンドウの下のペインを見てください．選択したメニュー項目の識別子に定義された値がwParamに渡されていることがわかるでしょう．もしウィンドウが小さくて全部が見られない場合は，右下のリサイズコーナーをドラッグするか[f･5]を押してウィンドウを拡大します．

Borland C++の問題点

Borland C++がWindowsアプリケーションを開発する環境として提供されるのであれば，IDEがWindowsアプリケーションでないのは惜しむべき点かもしれません．しかし，DOS用の慣れたFEPを使いたいこともありますし，DOSアプリケーションの開発とまったく同じユーザーインターフェースを使える点は決してデメリットではないでしょう．

むしろ，なによりも望まれるのは標準的なWindows用のクラスライブラリです．いまのところ，どんなウィンドウの性質をあらわすクラスも結局は自分で定義してやらなければなりません．C++はそうした手間を軽減するためには役立ちますが，その手間をなくしたいと思っているプログラマは少なくないでしょう．

幸いなことに，先日行われたOOP TOURの中でボーランドはObject Windows Libraryというクラスライブラリの開発を明言しました．これは，現在Turbo Pascal for Windowsで用意されているクラスライブラリをBorland C++用に変更したもので，おそらく本誌が発行される頃にはIBM-PC用のものが出荷されているのではないかと思います．いずれ，移植作業も行われるでしょう．Borland C++が無敵の開発環境と呼ばれるのはそれからかもしれません．

参考文献：
1.『プログラミングWINDOWS1，2』Charles Petzold著，エー・ピー・ラボ訳，アスキー出版局
2.『C++言語入門』Stephen C. Dewhurst，Kathy T. Stark共著，小山裕司監訳，アスキー出版局
3.『MS-WINDOWSソフトウェア開発キット　プログラマーズガイド』日本電気
4.『Borland C++　ユーザーズガイド』Borland International
5.『Borland C++ WINPRIMR .DOC』Borland International

Windowsソフトウェア一覧
TexTMS，198,000円，言語・開発ツール，ソニーテクトロニクス

特集 Windows
プログラムの作り方

クラスライブラリを使った オブジェクト指向プログラミング
Turbo Pascal for Windows

有澤雄志

Turbo Pascal for Windows(以下TPW)は，Borland社から発売されたWindowsアプリケーション開発環境である．これまで，Windowsアプリケーションを作る言語としては，MS-Cをはじめとした C言語が主流を占めていたが，TPWの登場によって，Pascalプログラマも Windows上でその力を発揮することができるようになった．CやC++を日常的に使っている Windowsプログラマは，「Pascalでなにができる?」と疑問をもつかもしれないが，Pascalの表現力はC，C++と比べても遜色のないものであり，C+SDKでできることはTPWでも可能である．さらにTPWでOWL (ObjectWindows Library)を使えば，C+SDKよりもはるかに楽にWindowsアプリケーションを作ることができる．今後，ますます複雑化するWindows/DOSプログラミングにおいてプログラマを救う鍵の1つかオブジェクト指向である．TPWとOWLは，クラスライブラリを用いたオブジェクト指向プログラミングを学ぶ最適の環境といえるだろう．

Object-Pascalとは

TPWでは，標準Pascalをオブジェクト指向的に拡張したObject-Pascalを扱うことができます．Object-Pascalに追加されたキーワードは，「object」「virtual」「construct」「destruct」そして「private」だけです．追加されたキーワードがこれしかないことからもわかるように，オブジェクト指向プログラミング言語としては単純な部類に入ります．

いまをときめくC++と比べると，

①多重継承のメカニズムが存在しない．
②カプセル化が弱い．
③Constructor/Destructorを明示的に呼ばなければいけない．

などの点で大きく異なります．さらに演算子オーバーライドができないなど，C++やSmalltalkのように本格的なオブジェクト指向プログラミング言語に比べると見劣りすると思うかもしれません．しかし，「単純=低機能」ではないのです．かえって，その分標準Pascalとの差が少ないので，オブジェクト指向プログラミングの経験の少ないプログラマでも気楽に取り組むことができます．

TPWのインストールと使い方

では，簡単にTPWのインストールとIDE(統合開発環境)の使い方を説明します．

TPWのインストールプログラムは，Windows下で動くようになっているので，TPWのフロッピーディスクの1枚目にあるINSTALL.EXEをWindowsのファイルマネージャから実行します．Windows APIの解説が含まれているヘルプファイルやリソース編集ツールをすべて展開すると，約10MBのディスク容量を消費します．

ファイルの展開が終わると，環境変数PATHに対する変更要求が表示されます．これはTPWの実行には影響しませんが，TPWからTDW(Turbo Debugger for Windows)を起動するときに必要になりますからメモしておきます．

これでインストールは終わりです．

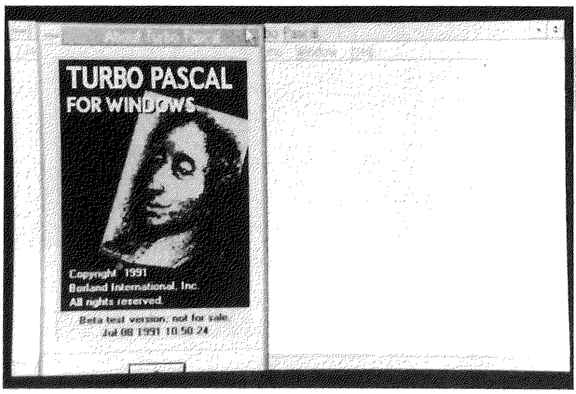


写真1
TPWのAbout

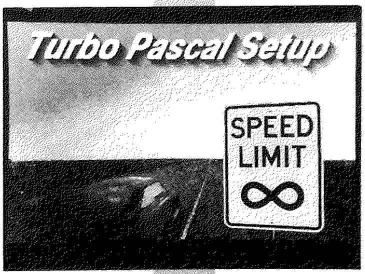


写真2　INSTALL.EXEの背景

INSTALL.EXEでTPWをインストールした場合は，ユニットやリソースの検索パス名などが自動的に設定されるので，とくにディレクトリ設定をする必要はありません．

TPWに内蔵されているエディタは，2種類のキーアサインを切り替えて使うことができます．1つはTurbo CやTurbo PascalでおなじみのWordStar風のキーアサインで，もう1つはWindowsの統一規約であるCUA(Common User Access)体系にしたがったキーアサインです．VZやMIFESの標準のキーアサインは前者に似ています．それに対して，Windowsの入出力行やメモ帳(NOTEPAD.EXE)では後者が使われています．この2つのモードは，TPWのメニューバーのOptionメニューからPreferencesダイアログを表示させ，CUAとAlternateのラジオボタンを選択することで設定できます．

IDEの特徴としては，32枚まで開くことのできる編集ウィンドウ，メモリの許す限り元に戻れるUndo機能などがあります．プログラムができあがったら，Makeメニューを選ぶか[F9]キーを押せば，ユニット同士の依存関係を調べてBuildしてくれます．DOS版のIDEと同じく，構文エラーがあればそのソースを表示し，エラー位置を反転して示します．

メニューからDebuggerを選ぶとTDWが起動します．TPWでは，Linker Optionで.EXEファイルにデバッグ情報を付けないように指定していた場合でも，TDWを起動する前に自動的にリンクしなおしてデバッグ情報を作りなおします．

デバッグが終われば，TDWを起動する直前のTPWの画面に戻ります．問題点を修正し，コンパイルしなおすためにTPWを抜ける必要はありません．編集中のすべてのファイルとそのカーソル位置，ファイル名や検索文字列のヒストリなどは完全に保存されています．

Pascalと Windows

Windows上の公式開発言語は（運の悪いことに）C言語です．そのため，Windows上のAPIはC言語で呼ばれることを前提として作られています．

CとPascalはいろいろな点で異なりますが，Windowsでプログラムを組む立場から見ると，文字列の表現方法の相違がもっとも大きな問題となるでしょう．Cの文字列は，通称ASCIZ文字列と呼ばれる最後に1バイトの“0”がついた文字の連なりです．これに対してPascalの文字列は，頭に文字列の長さが存在する文字の集合です．

この2つの形式をプログラマの手を煩わせずに共存させるために，TPWには従来のTurbo Pascalの基本型定義に加えて新しく“PChar”型が追加されています．これは，C言語の文字列と互換性があり，WindowsAPIを呼ぶときに使用できます．また，この型は0を下限とする文字の集合とも相互に代入可能となっています．

PChar型の変数に文字列定数を代入すると，コンパイラがその文字列の最後に0を自動的に追加します．よって，リスト1のように見かけ上同じ関数呼び出しでも，①では文字列'ABCDEFG'の後ろに0がついたものが自動的に作り出され，その先頭のポインタが手続きCStyleStringに渡されます．それに対して②では，先頭が文字列長の6，その後ろに'ABCDEFG'の各文字が格納された配列が手続きPStyleStringに渡されます．

LIST.1

```
procedure CStyleString( Str : PChar);
procedure PStyleString( Str : string );

begin
  CStyleString( 'ABCDEFG' );   { ① }
  PStyleString( 'ABCDEFG' );   { ② }
end.
```

LIST.2

TRAD.pas

```
{ "古典的" スタイルでの Hello! Window! プログラム。}

program TraditionalStyle;

uses WinTypes, WinProcs, Strings;

const
  AppName = 'Trad';

function WindowProc(Window: HWnd; Message, WParam: Word;
  LParam: Longint): Longint; export;
const
  msg = 'Hello! Windows!';
var
  dc : HDC;
  ps : TPaintStruct;
begin
  WindowProc := 0;
  case Message of
    wm_Paint: begin
      dc := BeginPaint(Window, ps);
      TextOut(dc, 0,0, msg, StrLen(msg));  { msg を画面に表示。}
      EndPaint(Window, ps); end;
    wm_Destroy: begin
      PostQuitMessage(0);
      Exit; end; end;
  WindowProc := DefWindowProc(Window, Message, WParam, LParam);
end;

procedure WinMain;
var
  Window: HWnd;
  Message: TMsg;
const
  WindowClass: TWndClass = (
    style: 0;
    lpfnWndProc: @WindowProc;
    cbClsExtra: 0;
    cbWndExtra: 0;
    hInstance: 0;
    hIcon: 0;
    hCursor: 0;
    hbrBackground: 0;
    lpszMenuName: AppName;
    lpszClassName: AppName);
begin
  if HPrevInst = 0 then begin
    WindowClass.hInstance := HInstance;
    WindowClass.hIcon := LoadIcon(0, idi_Application);
    WindowClass.hCursor := LoadCursor(0, idc_Arrow);
    WindowClass.hbrBackground := GetStockObject(white_Brush);
    if not RegisterClass(WindowClass) then Halt(255); end;
  Window := CreateWindow(
    AppName,
    'HelloWin',
    ws_OverlappedWindow,
    cw_UseDefault, cw_UseDefault,
    cw_UseDefault, cw_UseDefault,
    0, 0,
    HInstance,
    nil);
  ShowWindow(Window, CmdShow);
  UpdateWindow(Window);
  while GetMessage(Message, 0, 0, 0) do begin
    TranslateMessage(Message);
    DispatchMessage(Message); end;
  Halt(Message.wParam);
end;

begin
  WinMain;
end.
```

もちろん，既存の文字列操作機構はPChar型の文字列に対しては使えません．これを補うために，新しくStringsユニットが供給されています．このユニットを使えば，PChar型の文字列の代入／比較／検索などが簡単にできるようになります．もちろん，STRING型とPChar型との相互変換も用意されています．

LIST.3

WCRT.pas

```
{ WinCrt ユニットを使った "DOS" 風 Hello Window }
uses WinCrt;

begin
  WriteLn('Hello! Windows!');
end.
```

LIST.4

```
uses WinTypes, WinProcs, WinCrt;
var
  w : HWnd;
  dc : HDC;

begin
 InitWinCrt;        { ウィンドウを開く。 }
 . . .
 w := GetActiveWindow;
 dc := GetDC(w);    { この dc を使って GDI API を呼ぶことが出来る。}
 . . .
 ReleaseDC(dc);
end.
```

Traditional, WinCrt, OWL

伝統的なプログラミングスタイル

さて，ここで“HelloWin”プログラムを作ってみましょう．

TPWでWindowsプログラムを作る際には，「伝統的」「WinCRT」「OWL」という3通りのスタイルを選ぶことができます．

前にも書いたように，C言語で可能なことはPascalでも可能です．「伝統的」なスタイルとは，C言語で書かれたWindowsプログラムと同じことをPascalで記述することです．リスト2を見るとわかるように，1行文字列を出すために約70行もソースを記述しなければいけません．

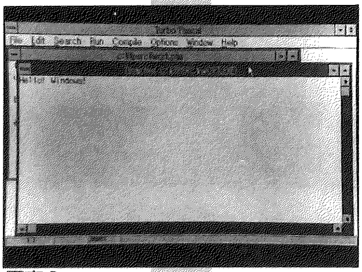


写真3　WinCRTの画面

このプログラムで肝心な部分は，22行目のTextOutのみです．残りの部分は表示を行うための下準備に過ぎません．こういう本質ではない部分，毎回同じことをしなければいけない部分こそ，ライブラリで吸収すべきです．

WinCRTによるプログラミングスタイル

WinCRTユニットは，このような不満に対する1つの解答です．WCRT.pas(リスト3)の最初の1行にあるUses　WinCrt;が，今までのWindowsプログラミングの手間の大部分をなくしてくれます．実際，このプログラムには本質的なコードしか書かれていません！

WinCrtユニットを使うことで，Windowsの画面の中に疑似的なテキストベースの画面が生成されます．標準では，DOSの画面に合わせて80×25文字の画面です．このユニットを使うことで，ウィンドウを通して標準Pascalで使っていたWrite文やRead文を使うことが可能になります．これは，既存のDOSアプリケーションの多くが，簡単にWindowsアプリケーションとして生まれ変わることを意味しています．

しかし，WinCrtが提供する疑似テキスト画面が，Windowsの管理するグラフィカルなウィンドウであることを忘れてはいけません．たとえば，リスト4のようにすれば，WinCrtを利用したプログラムの中で，Windowsが供給するすべてのGDI／描画APIが使えるのです．

TPWとWinCRTを使うと，その統合開発環境の使い心地と相まってインタプリタ感覚でWindowsAPIを1つずつ試していくことができます．GUIプログラミングを理解するには，実際にAPIを呼んで，何か画面に描いてみることが必要です．しかし，初心者が気楽に使える環境はなかなかありません．その意味で，TPWはWin-

Windows ソフトウェア一覧
Tiffany Plus，38,000円，グラフィックスデータコンバータ，レンガ

dowsプログラミングを理解するための近道といえるでしょう．

また，WinCRTのソースはすべて公開されているので，自分専用のWinCRTを作ることもできます．

OWLでのプログラミングスタイル

WinCRTは十分強力ですが，ライブラリとしては柔軟性にかけるところがあります．メニューを追加しようと考えただけで，WinCRTのソースを変更しなければなりません．もちろん，「伝統的」スタイルにおいては普通の作業です．土台となるソースをもってきて，タイトルを変え，新しいメッセージを用意してダイアログを追加し……．しかし，たいていの場合，変更点は一部分に限られます．本当に，そのたびにソースコードを1から変更しなければいけないのでしょうか？

TPWに付属するクラスライブラリであるOWLは，これらの不満を完全に解決してくれます．

リスト5，6を見てください．Win-CRTユニットを使ったWCRT.pasより行数は増えていますが，WCRT.

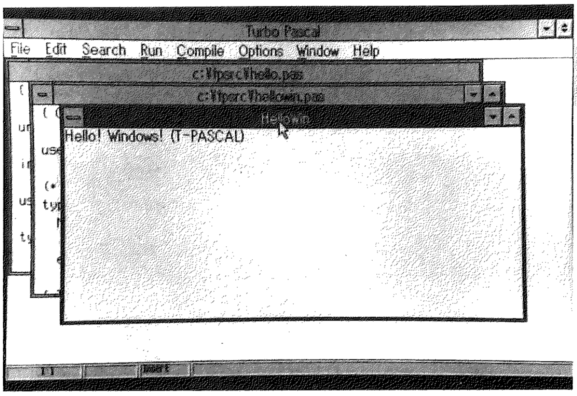


写真4

OWLの画面

pasと異なり，このプログラムは無限の可能性を内蔵しています．このあと，OWLを使いながらこのプログラムを拡張してみましょう．

Windowsでは周知の通り，各種の指令がメッセージとしてやってきます．Windowsプログラミングとは，「あるメッセージ」に対して「なにをするか」を記述すること，といっても過言ではありません．wm_Paintが与えられたら自分の責任領域を描画しなおす，というのはその一例です．

OWLでは，主要なメッセージに対して，それぞれメソッドが用意されています．これらのメソッドは，Object-Pascalのオブジェクトの継承機構によって，もっとも基本的なWindowsObjectから継承されます．プログラマは自分に必要なメッセージを受信するメソッドを作り，それに自分の行いたい動作を定義して新しいオブジェクトを作るわけです．「マウスをドラッグさせることによってウィンドウの大きさを変更する」ような，とくに変更しなくてもよいメッセージと動作の組み合わせは，祖先のメソッドの機能を継承してしまえばよいわけですから，何も記述する必要はありません．

LIST.5

Hello.pas

```
{ THelloWindow オブジェクトの定義。以後のプログラムで使用。}

unit Hello;

interface

uses WObjects, WinProcs, WinTypes;

type
  PHelloWindow = ^THelloWindow;
  THelloWindow = object(TWindow)
    procedure Paint(PaintDC: HDC; var PaintInfo: TPaintStruct); virtual;
  end;

implementation

uses Strings;

procedure THelloWindow.Paint(PaintDC: HDC; var PaintInfo: TPaintStruct);
const
  msg = 'Hello! Windows! (T-PASCAL)';
begin
  TextOut(PaintDC, 0,0, msg, strlen(msg) );
end;

end.
```

TApplication

では，もっとも小さいOWLのプログラムを紹介しましょう（リスト7）．このTAPP.pasは，TApplicationというオブジェクトを変更せずにそのまま使っています．もしTPW/TDWが手元にあるのなら，1行ずつステップ実行してみてください．Object-Pascalでは，オブジェクトを作ったときには，自分でConstructorを呼ばなければいけません．TApplicationのInitコンストラクタは，引数としてアプリケーションの名称を要求しています．ここで設定した名称は，ウィンド

LIST.6

HelloWin.pas

```
{ OWL 版  Hello Window。}

uses WObjects, WinTypes, WinProcs, Hello;

(* ****************************************************************** *)
type
  MyApplication = object(TApplication)
    procedure InitMainWindow; virtual;
  end;

{ THelloWindow オブジェクトは Hello.pas で定義されています。}
procedure MyApplication.InitMainWindow;
begin
  MainWindow := New(PHelloWindow, Init(nil, 'HelloWin'));
end;

(* ****************************************************************** *)
var
  app : MyApplication;

begin
  app.Init('HelloWindow');
  app.Run;
  app.Done
end.
```

ウのキャプションで見ることができます．とりあえず何も指定しないでおきます．

これでオブジェクトの初期化が終わり，画面にウィンドウが表示されました．Runを呼び出して，ユーザーのメッセージを解釈させます．実際の処理はTApplicationのデフォルトのメソッドが行います．

システムメニューで終了を選ぶと，Runメソッドが終了します．画面上からもウィンドウが削除されました．最後にDoneが呼ばれ，全体の後かたずけを行います．

TAPP.pasとHelloWin.pasとを見比べてください．TRAD.pasで長々と記述しなければいけなかった，どのプログラムにも共通な部分がTApplicationの中に隠されてしまっています．なにもしないのならこれで十分です．普通は，なにかを画面に表示したいはずなので，新しく表示用のウィンドウを作り，TApplicationに貼り込みます．

まずリスト5のように，TWindow型からTHelloWindow型を導出します．'Hello!Windows!'と表示させたいので，Paintメソッドを書き換えます．それ以外のウィンドウの属性は変更する必要がないのでこれだけの変更ですみます．THelloWindowは，後々使うので独立したユニットにしておきます．

次にリスト6のように，MyApplication型のオブジェクトを作ります．InitMainWindowメソッドを書き換えて，THelloWindow型のオブジェクトを新しく作り，MyApplicationのMainWindowフィールドに代入するように変更します．

これで，THelloWindowがMyApplicationの下で表示できるようになります．HelloWin.pasはこうして作られました．

リソースの作成とリンク

せっかく作ったプログラムが'Hello! Window!'だけではさびしいので，メニューとアイコンを追加してみましょう．Windowsでは，これらのデータをリソースという形態で管理します．リソースの作成については，前章の「Windowsでも威力を発揮するオブジェクト指向言語 Borland C++」を参照してください．TPWにもBC++同様Resource Workshopが付属しているので，複雑なダイアログボックスやビットマップも短時間に作成することができます．

現バージョンのTPW統合環境には，リソースコンパイラは内蔵されていません．RESファイルを作るためには，TPWのパッケージに付属しているMicrosoftのリソースコンパイラ(RC)，またはBorlandのResource Workshopのどちらかを使わなくてはなりません．どちらを使っても，同じリソースを作ることができますが，Windows環境下で編集／テストができるResource Workshopを使うほうが便利です．

LIST.7

TAPP.pas

```
{ TApplication。}
uses WObjects;

var
  aApp : TApplication;

begin
  aApp.Init(''); { アプリケーションの名称. }
  aApp.Run;
  aApp.Done;
end.
```


Windows ソフトウェア一覧
VJE-γ 1.0，18,000円，日本語FEP，バックス

LIST.9

```
procedure MenuFileAbout (var Msg : TMessage); virtual cm_First + cm_ABOUT;

procedure THelloWindow.MenuFileAbout (var Msg : TMessage);
var
  Dialog : PDialog;
begin
  Dialog := New(PDialog, Init(@Self, 'GHOST'));
  Application^.ExecDialog(Dialog);
end;
```

MS－CやBC＋＋でWindowsプログラムにリソースを組み込むには，最終的なEXEやDLLに対してRCを実行します．BC＋＋ではプロジェクト／トランスファー機構によって自動化が進んでいますが，TPWではもう一歩先に進みました．

{$R res-file }

と，Pascalのプログラム文中に書くだけで，TPWがリソースファイルをリンクしてくれます．$Rコンパイラ指令によるリソースファイルの取り込みは，各リソースのIDが衝突していない限り複数書けます．さらに，この指令はユニット内でも有効です．リソースとプログラムは，もともと密接に結び付いているものです．この2つの関係をプログラム中で直接指定できることは，ファイル管理の点でもかなりの利点を生むでしょう．

IconとMenuの追加

リソースが用意できたところで，まずIconを設定しましょう．前述のHelloWinを土台にします．

プログラムが「最小化」を選択されてIcon状態になっているとき，WindowsはそのプログラムGetWindowClassを呼び出すことでIconのBitMapを貰います．それをデスクトップ上に表示しているのです．OWLの標準状態では，システムで用意されているidi_Applicationアイコンを返しています．

勘のいい読者は，もうIconの変更方法に気づいたでしょう．そうです，GetWindowClassを新しいクラスで上書きしてしまえばよいのです．

リスト5のTHelloWindow型を定義しているところにリスト8のコードを挿入し，GetWindowClassとInitを追加します．

ここでは，'GHOST'という名称のIconを自分自身から（hInstanceはプログラム自身を示しています）読み出して，TWindow型から継承したAWndClass.hIconに保管しています．

Menuも同様です．Initを書き換え，Attr.Menuフィールドにメニューを保管するだけです．MyApplicationは変更しなくてもよいところに注目してください．絵と額縁を別々に管理することで，同じ額縁に異なる絵を飾ることができます．それと同じことがOWLに当てはまるのです．

MenuのIDに対する動作の記述

これで，HelloWinにメニューバーが追加され，最小化するとかわいいアイコンがデスクトップの左隅に表示されるようになりました．しかし，メニューをいろいろいじっても何も起きません．なぜならメッセージに対する動作をまだ何も記述していないからです．

メニューからAboutを選択することにより，Windows内部でIDM_ABOUTの識別子をもったwm_CommandメッセージがHelloWinプログラムに渡されます．これを受信

LIST.8

```
procedure THelloWindow.GetWindowClass(var AWndClass: TWndClass);
begin
  TWindow.GetWindowClass(AWndClass);
  AWndClass.hIcon := LoadIcon(HInstance, 'GHOST');
end;

constructor THelloWindow.Init(AParent: PWindowsObject; ATitle: PChar);
begin
  TWindow.Init(AParent, ATitle);
  Attr.Menu := LoadMenu(HInstance, 'GHOST');
end;
```

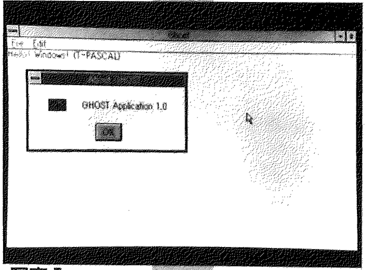


写真5　GHOSTのAbout画面

してアバウトダイアログボックスを表示させるためには，リスト9の最初の1行をTHelloWindow型の定義に追加し，メソッドの定義を追加します．

この変更により，'GHOST'という名称のダイアログリソースを読み込み，モードつきダイアログを生成します．

注意深い方なら，THelpWindowのMenuFileAboutのお尻に，変な記述があることに気がつかれたでしょう．これはTPWで拡張された構文で，メソッドをDynamic Method Table（以下DMT）のほうに置くように指示する機能をもちます．

Object-Pascalは，レートバインディングを実現するために，Virtual Method Table（以下VMT）をもっています．ここにメソッドの実体へのポインタを格納することで，コンパイルしたあとでも，実行するメソッドを切り替えることができるのです．

DMTはVMTを拡張したもので，1つ1つのメソッドにインデックスをふり付けることが可能となっています．じつは，このインデックスをWindowsのすべてのメッセージに対応させたことがOWLの核心なのです．

たとえば，「virtual wm_First + wm_Paint;」をメソッド定義の後ろに追加しておけば，wm_Paintがきたときにそのメソッドが呼ばれます（wm_Paintに関していえば，重要なメッセージなのでPaintという名のメソッドがすでに定義されています）．

一般のWindowsのメッセージは，wm_First + wm_xxxxxの形で定義し，wm_Commandメッセージはさらに細分化され，cm_First + cm_xxxxの形で定義ができます．

「伝統的」スタイルのWindowsプログラムでは，どんな言語を使おうとも，メッセージを仕分けるための条件分岐の山ができてしまいます．ときとしてそれは，数画面におよぶ巨大な関数を生み，プログラムの保守性を著しく損なっていました．

OWLは，Windows自身がもつオブジェクト指向の部分を，Object-Pascal自体の拡張により自然な形で取り込んだのです．

さて，今まで述べてきたHelloWinの変更点を1つにまとめ，いくつか機能を追加したものをリスト10に

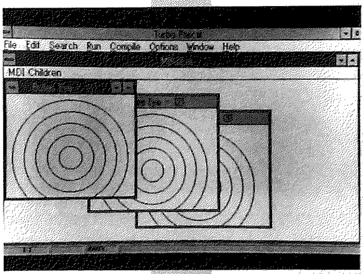


写真6　BullsEyeとMDI

LIST.10

GHOST.pas

```
{ メニューとアイコンを追加。BC++ の GHOST.cpp と比べてみてください。}
uses WObjects, WinTypes, WinProcs, Hello;

{$R Ghost.res}

(* ******************************************************************** *)
const
  cm_NEW       = 101;
  cm_OPEN      = 102;
  cm_SAVE      = 103;
  cm_SAVEAS    = 104;
  cm_ABOUT     = 108;
  cm_EXIT      = 109;
  cm_UNDO      = 110;
  cm_CUT       = 111;
  cm_COPY      = 112;
  cm_PASTE     = 113;
  cm_DELETE    = 114;

(* ******************************************************************** *)
type
  PGHelloWindow= ^TGHelloWindow;
  TGHelloWindow= object(THelloWindow)
    constructor Init(AParent: PWindowsObject; ATitle: PChar);
    procedure GetWindowClass(var AWndClass: TWndClass); virtual;
    procedure DefWndProc(var Msg : TMessage); virtual;
    procedure MenuFileAbout (var Msg : TMessage); virtual cm_First + cm_ABOUT;
    procedure MenuFileExit (var Msg : TMessage); virtual cm_First + cm_EXIT;
  end;

constructor TGHelloWindow.Init(AParent: PWindowsObject; ATitle: PChar);
begin
  THelloWindow.Init(AParent, ATitle);
  Attr.Menu := LoadMenu(HInstance, 'GHOST');
end;

procedure TGHelloWindow.GetWindowClass(var AWndClass: TWndClass);
begin
  THelloWindow.GetWindowClass(AWndClass);
  AWndClass.hIcon := LoadIcon(HInstance, 'GHOST');
end;

procedure TGHelloWindow.DefWndProc(var Msg : TMessage);
begin
  if (Msg.Message = wm_command) and (Msg.wParam >= cm_New) then begin
    MessageBox(Msg.Receiver, 'This feature not supported', Nil, MB_ICONASTERISK + MB_OK); end
  else TWindow.DefWndProc(Msg);
end;

procedure TGHelloWindow.MenuFileAbout (var Msg : TMessage);
var
  Dialog : PDialog;
begin
  Dialog := New(PDialog, Init(@Self, 'GHOST'));
  Application^.ExecDialog(Dialog);
end;

procedure TGHelloWindow.MenuFileExit (var Msg : TMessage);
begin
  CloseWindow;
end;

(* ******************************************************************** *)
type
  MyApplication = object(TApplication)
    procedure InitMainWindow; virtual;
  end;

procedure MyApplication.InitMainWindow;
begin
  MainWindow := New(PGHelloWindow, Init(nil, 'Ghost'));
end;

(* ******************************************************************** *)
var
  Ghost : MyApplication;

begin
  Ghost.Init('Ghost/Pascal');
  Ghost.Run;
  Ghost.Done
end.
```


Windowsソフトウェア一覧
Windows BACKGAMMON，9,800円，ゲーム，レンガ

Ghost.pasとして掲載しておきます.

MDI(Multiple-Document-Interface)

まず, TBullsEyeというオブジェクトを作ります(リスト11). Paintしかしないので, THelloWindowオブジェクトと似通っていますが, 文字を書くかわりに5つの同心円を描画するようになっています. BullsEyeもTWindowから作られているので, HelloWinやGhostでTHelloWinオブジェクトの代わりに使うことができます.

前に絵と額縁のたとえをしましたが, こんどは額縁のほうを変更してみましょう. 今までは, TWindowから導出したオブジェクトをそのままTApplicationに貼り付けていましたが, こんどは複数のTWindowを貼り付けることのできるTMDIWindowを間にはさんでみます.

LIST.11

BullsEye.pas

```
{ TBullsEye の定義。}
Unit BullsEye;

interface

uses WObjects, WinTypes, WinProcs;

type
  PBullsEye = ^TBullsEye;
  TBullsEye = object(TWindow)
    procedure Paint(PaintDC: HDC; var PaintInfo: TPaintStruct); virtual;
  end;

implementation

procedure TBullsEye.Paint(PaintDC: HDC; var PaintInfo: TPaintStruct);
var
  i : integer;
begin
  SelectObject(PaintDC, GetStockObject(Null_Brush));
  for i := 1 to 5 do Ellipse(PaintDC, (110 - 20 * i), (110 - 20 * i),
                                      (110 + 20 * i), (110 + 20 * i));
end;

end.
```

MBullEye.pas(リスト12)をHelloWinと比べてみてください. たったこれだけで, Multiple-Document-Interfaceにのっとった画面ができてしまうのです. OWLのほうで定義されているメニューから'Create'を指定すれば, いくらでも'Bulls Eye'ウィンドウができていきます. カスケード/タイリングも可能です.

LIST.12

MBullEye.pas

```
{ MDI のサンプル。BullsEys と Hello ユニットを使います。}

{$R MDIAPP.RES}
{$R Ghost.res}

uses WObjects, WinProcs, WinTypes, BullsEye, Hello;

(******************************************************************)
type
  PMBull = ^TMBull;
  TMBull = object(TMDIWindow)
    Bull, Hello : word;
    constructor Init(ATitle: PChar; AMenu: HMenu);
    function InitChild : PWindowsObject; virtual;
  end;

constructor TMBull.Init(ATitle: PChar; AMenu: HMenu);
begin
  TMDIWindow.Init( ATitle, AMenu );
  Bull := 0;
  Hello := 0;
end;

function TMBull.InitChild: PWindowsObject;
var
  NameBuf : array[0..30] of char;
begin
  if Random(2) = 0 then begin
    Inc(Bull);
    wvsprintf(NameBuf, 'Bulles Eye - (%d)', Bull);
    InitChild := New(PBullsEye, Init(@self, NameBuf)); end
  else begin
    Inc(Hello);
    wvsprintf(NameBuf, 'HelloWin - (%d)', Hello);
    InitChild := New(PHelloWindow, Init(@self, NameBuf)); end;
end;

(******************************************************************)
type

  TMDIApplication = object(TApplication)
    procedure InitMainWindow; virtual;
  end;

procedure TMDIApplication.InitMainWindow;
begin
  MainWindow := New(PMBull, Init('MBullEye',
    LoadMenu(HInstance, 'MDIMenu')));
end;

(******************************************************************)
var
  MyApplication: TMDIApplication;

{ MyApplicationの実行 }
begin
  Randomize;
  MyApplication.Init('Multi Bulls Eye.');
  MyApplication.Run;
  MyApplication.Done;
end.
```

クリップボード

BullsEyeで書いた絵を, こんどはクリップボードに転送してみましょう. BullCopy.pas (リスト13) は,

Ghost.pasを土台にしてTHelloWindowオブジェクトのかわりに，TBullEyeオブジェクトを使ったものです．

メニューバーからEdit/Copyが選択されると，cm_First + cm_Copyのインデックスをもったメソッドが TBullWindowの中から探されます．この場合は存在するので，MenuEditCopyメソッドが実行されます．

ウィンドウ自身の大きさを取得し，aBitMapに転送しています．

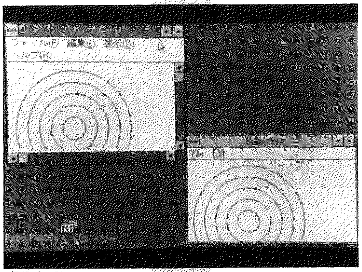


写真7　BullCopyとクリップボード

おもにOWLの使い方を説明してきましたが，TPW+OWLを使えばWindowsアプリケーションを作るための労力を，かなり削減できることがおわかりいただけたと思います．

誤解のないように記述しておきますが，OWLは「Windowsを知らなくてもプログラムが書ける」ライブラリではありません．

DOSを知らなければ，ディスクにファイルを書き出すことができないように，Windowsを知らなければウィンドウに絵を描くことはできません．しかし，絵を描くこと以外にしなければいけないことが，あまりにも多すぎました．OWLは，Windowsのもつ雑多な部分を吸収し整理した環境を供給するライブラリなのです．

LIST.13

BullCopy.pas

```
{ クリップボードへのコピー }
uses WObjects, WinTypes, WinProcs, Strings, BullsEye;

{$R Ghost.res}

(* ****************************************************************** *)
const
  cm_NEW        = 101;
  cm_OPEN       = 102;
  cm_SAVE       = 103;
  cm_SAVEAS     = 104;
  cm_ABOUT      = 108;
  cm_EXIT       = 109;
  cm_UNDO       = 110;
  cm_CUT        = 111;
  cm_COPY       = 112;
  cm_PASTE      = 113;
  cm_DELETE     = 114;

(* ****************************************************************** *)
type
  PBullWindow = ^TBullWindow;
  TBullWindow = object(TBullsEye)
    constructor Init(AParent: PWindowsObject; ATitle: PChar);
    procedure GetWindowClass(var AWndClass: TWndClass); virtual;
    procedure DefWndProc(var Msg : TMessage); virtual;
    procedure MenuFileAbout (var Msg : TMessage); virtual cm_First + cm_ABOUT;
    procedure MenuFileExit (var Msg : TMessage); virtual cm_First + cm_EXIT;
    procedure MenuEditCopy (var Msg : TMessage); virtual cm_First + cm_COPY;
  end;

constructor TBullWindow.Init(AParent: PWindowsObject; ATitle: PChar);
begin
  TWindow.Init(AParent, ATitle);
  Attr.Menu := LoadMenu(HInstance, 'GHOST');
end;

procedure TBullWindow.GetWindowClass(var AWndClass: TWndClass);
begin
  TWindow.GetWindowClass(AWndClass);
  AWndClass.hIcon := LoadIcon(HInstance, 'GHOST');
end;

procedure TBullWindow.DefWndProc(var Msg : TMessage);
begin
  if (Msg.Message = wm_command) and (Msg.wParam >= cm_New) then begin
    MessageBox(Msg.Receiver, 'This feature not supported', Nil, MB_ICONASTERISK + MB_OK); end
  else TWindow.DefWndProc(Msg);
end;

procedure TBullWindow.MenuFileAbout (var Msg : TMessage);
var
  Dialog : PDialog;
begin
  Dialog := New(PDialog, Init(@Self, 'GHOST'));
  Application^.ExecDialog(Dialog);
end;

procedure TBullWindow.MenuFileExit (var Msg : TMessage);
begin
  CloseWindow;
end;

procedure TBullWindow.MenuEditCopy (var Msg : TMessage);
var
  R : TRect;
  SizeX, SizeY : integer;
  DC, Buffer : HDC;
  aBitMap : HBitMap;
begin
  GetClientRect(hWindow, R);
  SizeX := (R.Right - R.Left);
  SizeY := (R.Bottom - R.Top);
  DC := GetDC(hWindow);
  Buffer := CreateCompatibleDC(DC);
  aBitMap := CreateCompatibleBitmap(DC, SizeX, SizeY);
  if (aBitMap <> 0) then begin
    SelectObject(Buffer, aBitMap);
    if BitBlt(Buffer, 0, 0, SizeX, SizeY, DC, 0, 0, SrcCopy) then begin
      if OpenClipboard(HWindow) then begin
        EmptyClipboard;
        SetClipboardData(cf_Bitmap, aBitMap);
        CloseClipboard; end; end; end;
  DeleteDC(Buffer);
  ReleaseDC(hWindow, DC);
end;

(* ****************************************************************** *)
type
  MyApplication = object(TApplication)
    procedure InitMainWindow; virtual;
  end;

procedure MyApplication.InitMainWindow;
begin
  MainWindow := New(PBullWindow, Init(nil, 'Bulles Eye'));
end;

(* ****************************************************************** *)
var
  Ghost : MyApplication;

begin
  Ghost.Init('Bulles Eye Application');
  Ghost.Run;
  Ghost.Done
end.
```


Windows ソフトウェア一覧
WINDOWS日本語3270PC/G, 228,000円, 通信ソフト, 丸紅ハイテック・コーポレーション

ビジュアルなプログラミング環境を実現する VisualBASIC

海老原浩之

PROGRAMMING

Windows

この夏に米国で発売された VisualBASIC が評判をよんでいる. MS-Windows Ver.3.0で動作する環境を備えた Windows プログラム専用の開発言語である. この VisualBASIC は, プログラミングが容易にでき, 価格も安いというのが特徴である. 米国では, 発売以来すごい売れ行きで, 最初のころは手に入れるのに2, 3週間待ちという状態が続いたという. VisualBASIC 最大の特徴は, エディタでプログラミングしてコンパイルし, その後に Windows 上でプログラムを実行するようなめんどうがなく, すべての作業を Windows 上で, しかも Visual の名が示すように, 見たイメージのままにプログラミングできる環境にある. 実際, MS-C Ver.6.0+SDK で Windows のプログラミングをするよりも, はるかに簡単に Windows のプログラムを作成できる. ここでは, この VisualBASIC の概要と簡単なプログラムの実例を示す.

プログラム作成ツールの概要

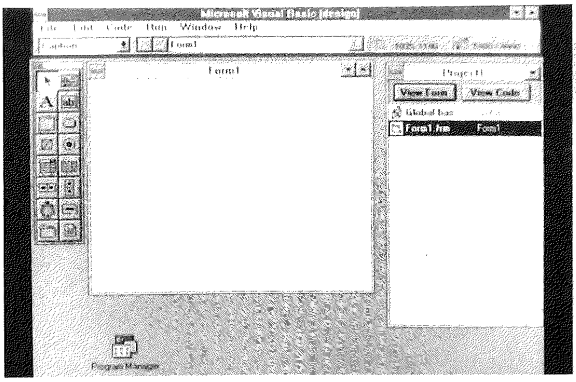


写真1
VisualBASICの起動画面

写真1を見てもらいたい．VisualBASICを実行すると最初に現れる画面である．写真1を見てもわかるように，最初は4つのウィンドウが開いている．この状態からプログラムの作成を開始していく．

VisualBASICでは，プログラムの中で使用するウィンドウの1つ1つをフォームと呼んでいる．VisualBASICによるプログラミングは，このフォームにスクロールバーやボタン，テキスト入力をする部分など(これらをまとめてコントロールと呼ぶ)を貼り付け，さらにこのコントロールにプロパティ(属性)を設定していくことで行われる．

PROPERTIES BAR

VisualBASICの本体である．この中から各種のメニューを選んで作業を進めていくことになる．プログラムを開発していく上で，中枢的な役割を果たす部分であるPROPERTIES BARには，次の2つが備えられている．

①プロパティリストボックス
②セッティッングボックス

①には，各TOOLごとに設定可能なプロパティの一覧が表示される．ここから使用するプロパティを選択して使っていくわけである．

使用するプロパティが決まったら，②でプロパティに対してどのようなことをするのかをセットする．たとえば，Captionというプロパティを設定したら，好きな文字を入れることができるし，Pictureというプロパティをセットすれば，好きなBMP画像やICONを読み込むことができる．

TOOL BOX

Windowsのアプリケーション開発に必要なツールが集合しているウィンドウである．この中から使いたい機能を選んで，フォームに貼り付けていくわけだ．各ツールは，図2のような機能をもっている．

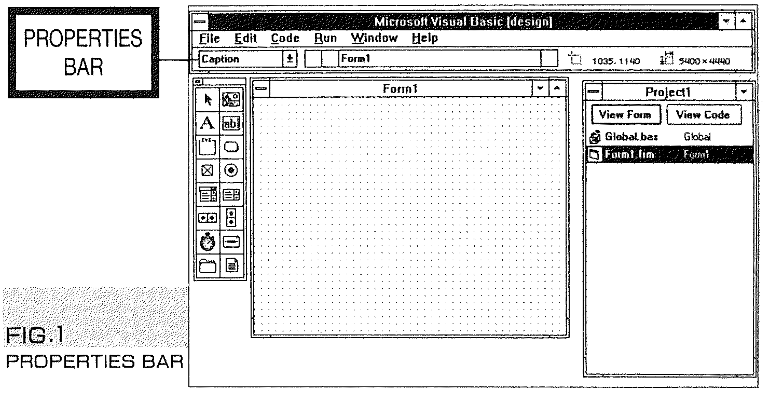
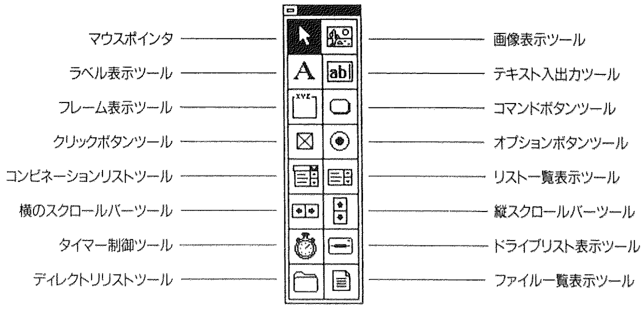


FIG.1
PROPERTIES BAR

FIG.2
TOOL BOX



Windowsソフトウェア一覧
WINTERM，28,000円，通信ソフト，日本電気コンピュータシステム

FORM WINDOW

このFORM WINDOWが，実際のアプリケーションの画面フォームである．ここに先ほどのTOOL BOXのツール類を貼り付けて，作成しようとするアプリケーションの画面の形を整えていくわけである．ただし，ツール類を貼り付けただけでは，完全なプログラムとしては動作しないので，各ツールをプログラミングによって制御したり，個々のツールを関連づけさせる必要がある．

PROJECT WINDOW

さまざまなフォームを使って1つのアプリケーションを作成した場合に，作成したすべてのフォームを管理するウィンドウである．これは，各アプリケーションごとに必ず1つ作成される．

VisualBASICの特徴

VisualBASICでは，プログラミング中でも各TOOLごとのヘルプやメソッド関数，通常の関数などが参照可能である．使用したいTOOLのボタンをマウスでクリックして[F1]キーを押せば，対応したTOOLのヘルプをすぐ呼び出せるし，プログラミング中の関数の使用法について知りたければ，関数にカーソルをおいて[F1]キーを押すと関数の説明を参照できる．

このONLINE HELPは，Quick BASICなどのONLINE HELPよりも便利である．なぜなら，高解像度デ

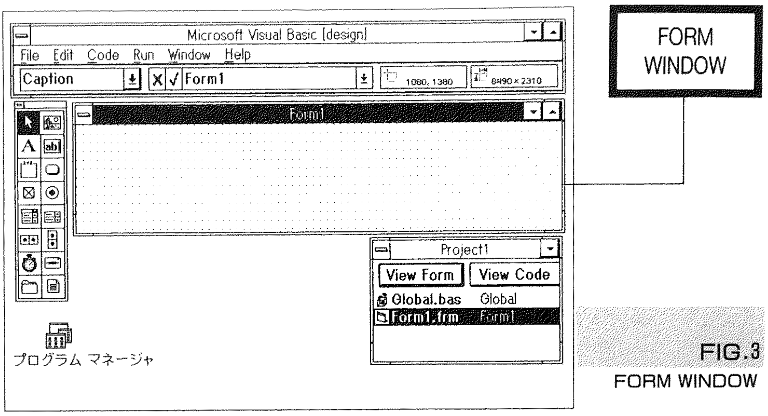


FIG.3
FORM WINDOW

CHART.1
プロパティ一覧

- ActiveControl
- ActiveForm
- Alignment
- Archive
- AutoRedraw
- AutoSize
- BackColor
- BorderStyle
- Cancel
- Caption
- Checked
- ControlBox
- CtlName
- CurrentX
- CurrentY
- Default
- DragIcon
- DragMode
- DrawMode
- DrawStyle
- DrawWidth
- Drive
- Enabled
- FileName
- FillColor
- FillStyle
- FontBold
- FontCount
- FontItalic
- FontName
- Fonts
- FontSize
- FontStrikethru
- FontTransparent
- FontUnderline
- ForeColor
- FormName
- hDC
- Height
- Hidden
- hWnd
- Icon
- Image
- Index
- Interval
- LargeChange
- Left
- LinkItem
- LinkMode
- LinkTimeout
- LinkTopic
- List
- ListCount
- ListIndex
- Max
- MaxButton
- Min
- MinButton
- MousePointer
- MultiLine
- Normal
- Page
- Parent
- Path
- Pattern
- Picture
- ReadOnly
- ScaleHeight
- ScaleLeft
- ScaleMode
- ScaleTop
- ScaleWidth
- ScrollBars
- SelLength
- SelStart
- SelText
- SmallChange
- Sorted
- Style
- System
- TabIndex
- TabStop
- Tag
- Top
- Value
- Visible
- Width
- WindowState

CHART.2
メソッド一覧

- AddItem
- Circle
- Clear
- Cls
- Drag
- EndDoc
- GetData
- GetFormat
- GetText
- Hide
- IsHidden
- Line
- LinkExecute
- LinkPoke
- LinkRequest
- LinkSend
- NewPage
- Point
- Print
- PrintForm
- Pset
- RemoveItem
- Scale
- SetData
- SetFocus
- SetText
- Show
- TextHeight
- TextWidth

ィスプレイでVisualBASICを使った場合，ヘルプウィンドウと実際にプログラミング作業をしているウィンドウとの重なりが少なく，ヘルプの内容とプログラミング画面を同時に見ることができるからである．これは，Windows環境ならではの優れた特徴といえる．

また，PROPERTIES BARのヘルプメニューにTutorialがあり，VisualBASICを学ぶことができる．VisualBASICのデモのつもりで見てもいいし，マニュアルを最初から読むよりはわかりやすいかもしれない．

VisualBASICの基本的な文法や関数などはQuickBASICと同じであるが，プログラミングの発想はまったく違ったものである．DOSのフルスクリーンで動作するアプリケーションを作成する場合，プログラミングのスタイルは基本的に上から下へ流れるようなものになる．しかし，Windowsのプログラミングスタイルは，イベントが発生してからそのイベントに対応した処理を行うというもので，上から下へ流れるような感じのものではない．

また，プロパティと呼ばれる属性やメソッド関数，VisualBASIC用に追加された関数なども覚える必要がある．

これらを身につけてしまえば，QuickBASICを使ったことがある人ならVisualBASICを使いこなすのは簡単だろう．さらに，今までMS-C+SDKでプログラミングをしてきた人は，あまりにも簡単にWindowsプログラムの開発ができることに驚きを覚えるだろう．参考までに，プロパティとメソッドの一覧を表1，2に示す．

VisualBASICの動作環境

VisualBASICは，米マイクロソフトが英語版Windows向けに販売している製品であるため基本的に英語環境で動作する．

VisualBASICを使用するには，以下のような環境が必要となる．

- MS-Windows 3.0もしくはそれ以上，動作モードはスタンダードモードか386エンハンストモード
- 80286以上のCPUが搭載されたパソコン
- 1MB以上のRAM
- CGA，EGA，VGA，8514/A，Hercules graphicsとコンパチブルなビデオグラフィックスカード
- マウス
- 3.5もしくは5インチFDDと空き領域が5MB以上あるHDD

上記は，IBM PC/AT（互換）もしくはIBM PS/2における動作環境であるが，実際には日本語MS-Windows Ver.3.0でも動作する．ただし，表示などの点で多少の不具合がある．実際に筆者は，以下の環境で動作確認をしている．

IBM PS/55 5551V0B

- RAM 16MB
- HDD 120MB
- IBM DOS Ver.J4.07/V
- 日本語MS-Windows Ver.3.01

NEC PC-9801RA21

- RAM 5.6MB

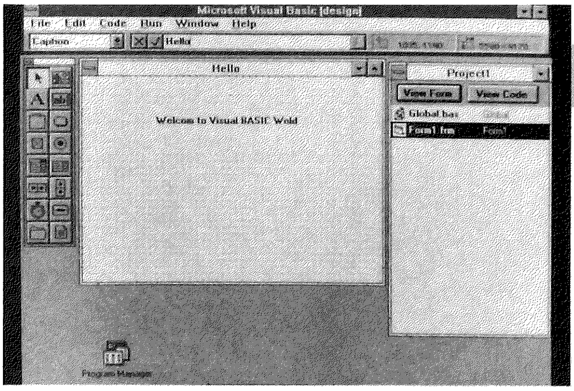


写真2
IBM PS/55上のVisualBASIC

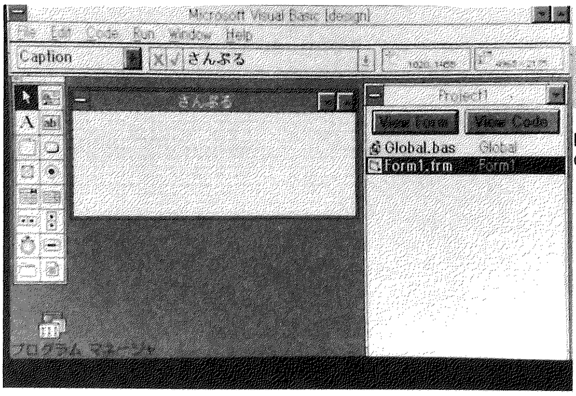


写真3
NEC PC-9801RA21上のVisualBASIC

- HDD 170MB
- MS-DOS Ver.3.30C
- 日本語MS-Windows Ver.3.00

写真2がIBM　PS/55，写真3がNEC　PC-9801RA21上のものである．

VisualBASICの開発ツール

VisualBASICには，別売されているツールが2つある．Control Development　KitとHELP　Compilerである．この2つのツールに関して簡単に説明する．

Control Development Kit（CDK）

これは，VisualBASICにあるTOOL　BOXに追加する新しいツールを作成するための開発キットである．これを使うためには，CDK以外にMS-C　Ver.6.0とSDKが必要となる．MS-C+SDKでWindowsのプログラムを作成した経験がないと，CDKを使って新しいツールを開発するのは難しいだろう．

実際に，TOOL BOXにCDKで作成された部品が追加された状態が写真4である．TOOL BOXに追加する方法は簡単で，メニューのFILEを選び，プルダウンメニューの中からAdd File...を選択するだけだ．このとき，拡張子が.VBXとなっているものがTOOL　BOXを拡張する部品ファイルである（写真5）．

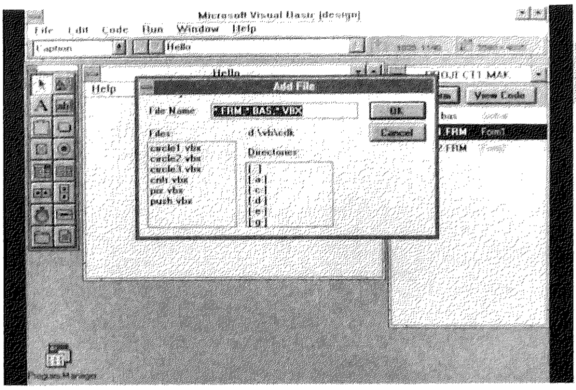


写真5
部品ファイル

HELP Compiler

VisualBASICには，HELP Compilerが付属していないので，ハイパーリンク式のWindows　HELPを作成するためには，別売されているHELP Compilerを買う必要がある．すでにSDKをもっている人は，SDKにHELP　Compilerが付属しているのでそれを使えば問題はない．

プログラムの作成

では，実際に簡単なプログラムを作成してみよう．今回は，紙面の関係上非常に簡単なプログラムとする．内容は，文字をウィンドウの中に表示し，メニューの選択によってダイアログボックスを表示するというものだ．しかし，これだけでもMS-C+SDKではたいへんな手間がかかる．VisualBASICになれてしまえば，この程度のプログラムは10分たらずでできてしまう．

では，順を追って説明していくことにする．

ウィンドウに文字を表示させる

有名な"Hello WORLD"と表示するだけのプログラムも，通常のDOSであれば非常に簡単であるが，これがWindowsとなれば単に表示するだけでもめんどうなものである．文字を表示するだけなのだが，MS-C+SDKではWindowsの各種制御をいちいち記述しなければならないため，呪文

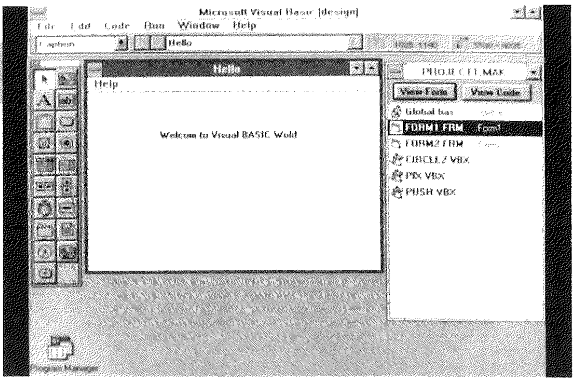


写真4
TOOL　BOXにCDKで作成された部品が追加された状態

のような前置きが非常に多くなる.

これに対して VisualBASICの場合, 単に文字を表示するだけならプログラムを組む必要はない. 先ほど説明した TOOL BOX の中からラベル表示ツールのボタンをマウスでクリックして, フォームに文字を書き込みたい領域をマウスで指定するだけである. そのあとは, プロパティリストボックスの Captionの右隣にあるセッティングボックスにマウスをもっていき, ここをクリックして好きな文字の入力をすればよいだけである(写真6).

また, ウィンドウに表示されるタイトルも同様にして簡単に変更できる. タイトルと表示する文字を入れ終わったら, PROPERTIES BARにあるメニューから FILEを選び, Make EXE Fileを実行すれば, 文字表示をするだけのプログラムができあがる.

メニューの作成

次にメニューの作成について説明する. このメニューの作成もビジュアルな状態で簡単に作ることができる. PROPERTIES BARにある Window を選び, プルダウンメニューから Menu Design Windowを選択すると, 写真7のようなメニューデザインウィンドウが表示されるので, ここでメニューを作成する.

ここで, HELP と ABOUT というメニューが出るように定義してみよう. Captionと書かれているボックスにメニュー名を入れると, その名前が実際にウィンドウのメニューとして表示される. 次に CtlNameを入れる. これは, プログラミングをする上での定義名称といったものだ. プログラミングにあたっては, ここで定義した名称を使うことになる.

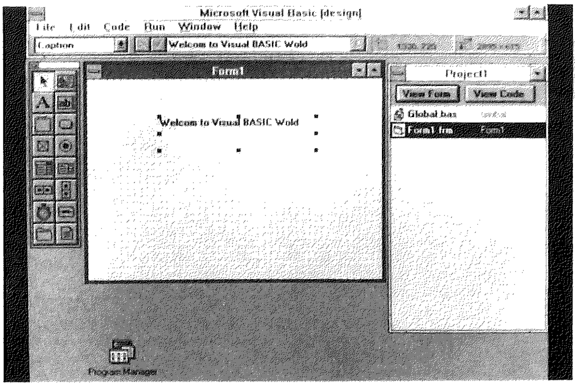


写真6

表示する文字を入力する

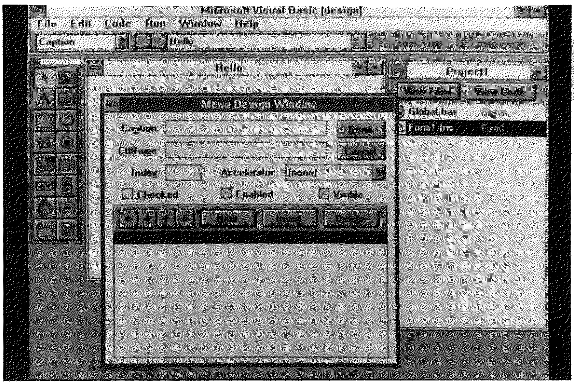


写真7

メニューデザインウィンドウ

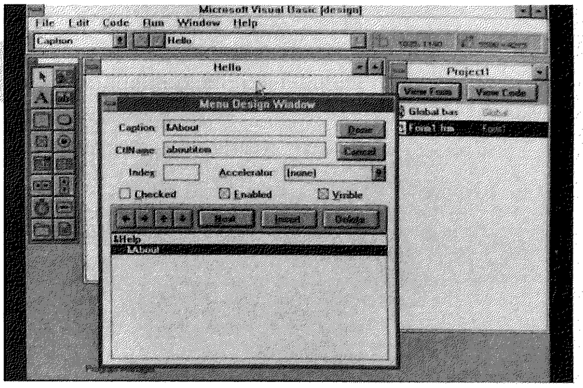


写真8

定義されたメニュー

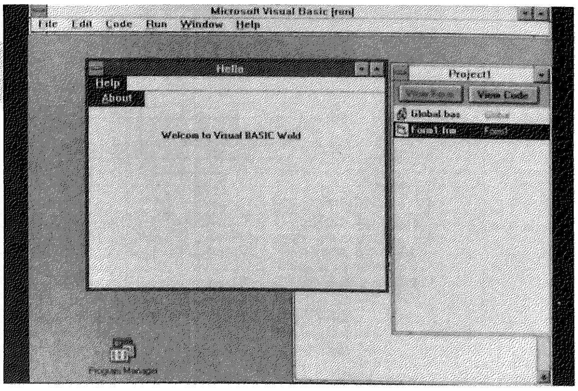


写真9

プログラムの実行

Windows ソフトウェア一覧
スマイルクラブ・情報室, 95,000円, 電子ファイリングシステム, ハイソフト

次に，字下げをして表示するメニューを作る．まず，Nextと書かれているボタンをマウスでクリックするとHELPのところにあったバーが1段下に移る．そこで右矢印をマウスでクリックすると字下げメニューになるので，あとはCaptionとCtlNameを入れるだけだ．

実際にメニューを定義した状態が写真8である．このメニューの状態でよければDoneをマウスでクリックして完成である．

また，一度作成したメニューに新たなメニューを追加したり，不必要なメニューを削除したりすることも，Insert(挿入)やDelete(削除)を選択することで簡単にできる．

さて，ここでプログラムを実行してみよう．PROPERTIES BARからRUNを選び，プルダウンメニューからSTARTを選択すると，ここまで作成したプログラムを実行できる．実行した状態が写真9である．

当然，STARTで実行したときにまちがいがあれば，エラー箇所でプログラムはストップするが，この段階ではプログラミングをいっさいしていないので，エラーが表示されることはない．プログラムを終了させるには，先ほどのプルダウンメニューからENDを選択すれば元の状態に戻る．

ダイアログボックスの作成

このHELLOプログラムに，プルダウンメニューからAboutを選択すると表示されるダイアログボックスを付け加えてみよう．VisualBASICでは，ダイアログボックスも新規にフォームウィンドウを表示させて作成する．PROPERTIES BARのメニューからFileを選び，プルダウンメニューからNew Formを選択すれば，新しいフォームが表示される(写真10)

先ほどと同じ要領で，Aboutを表示するダイアログボックスを作成する．FORM2のフォームをマウスで選択してタイトルや文字を書き込み，次に確認のボタンを付ける．まず，TOOL BOXの中からボタンツールを選び，フォーム2にマウスでボタンの大きさを指定して貼り付ける．大きさの指定は，文字領域を設定するときと同じである．実際にボタンを付けた状態を写真11に示す．

ボタンの中にCommand1と表示されているが，これはPROPERTIES BARのCaptionの隣に表示されているCommand1に，自分の好きな文字を書き込むことで自由に変えることができる．ここでは，「OK」と変えてみる．これで一応ダイアログボックスの形になった．

部品の関連付け

ここまでに作成したフォームやツールを，プログラミングによって関連付けてみよう．まず，1つ前に作成したダイアログボックスについてプログラミングをしてみる．

先ほど作成した確認ボタンをマウスでダブルクリックすると，form2.frmというタイトルのコードウィンドウが表示される．すでにコードウィンドウには，イベント処理のサブルーチンが定義されているので，その中にプログラムを追加するだけである．OKボタンをマウスでクリックすると，ダイアログボックスが消えるようにプログラミングするには以下のように記述する．

```
form2.hide
```

このプログラミングによって，OKボタンがマウスで押されると，自動的にダイアログボックスがクローズされる．form2.Hideの意味は，「form2というウィンドウを見えなくする」である．つまり，一連の処理の流れはリスト1のようになる．

プログラミングが終われば，プログラミングをしていたform2.frmのコードウィンドウをクローズする．

次に，先ほど作成したメニューとダイアログボックスを関連付けてみよう．Helpをマウスでクリックしてプルダウンメニューを表示させ，そこでAboutをマウスでダブルクリックしてみる．すると今度は，form1.frmとタイトル表示されたコードウィンドウが表示される．あとはform2.frmと同様にプログラミングするわけである．Aboutがマウスでクリックされたら，Aboutのダイアログボックスを表示するようにするには，以下のように記述する．

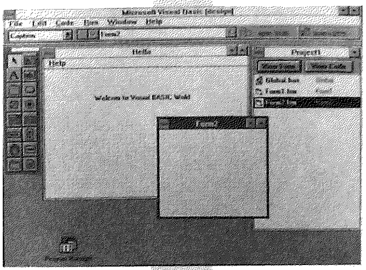


写真10　新規フォームの作成

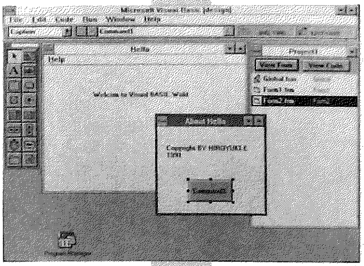


写真11　ダイアログボックスにボタンを付ける

Windows ソフトウェア一覧
ビジネスコレクション1，15,000円，グラフィックデータ集　サムシンググッドアルダス事業部

```
form2.Show
```

form2.Showの意味は，「form2というウィンドウを表示する」である．処理の流れはリスト2のようになる．

あとは，form1.frmのコードウィンドウをクローズしてプログラミングは終了である．

これで，簡単ながら今まで作成したフォームとボタンツールを押すといった処理すべての関連付けが完了した．では，実際にPROPERTIES BARのRUNでプログラムを実行させてみよう(写真12)．

簡単な例ではあるが，Visual-BASICのプログラム作成がいかに容易かわかってもらえたと思う．これだけのことをMS-C+SDKで行おうとすれば，こうはいかない．Visual-BASICのすばらしさは，まさにこの“容易さ”にあるのである．

アイコンの処理

最後に，完成したHelloプログラムにアイコンを付加してみよう．

プロパティリストボックスの横にある下矢印をマウスでクリックし，プロパティを一覧表示させる．その中にIconというプロパティがあるので，それをマウスで選択すると横にあるセッティングボックスに(Icon)と表示される．ここで[...]ボタンをマウスでクリックするとLoad Iconというウィンドウが表示されるので，アイコンがあるディレクトリを指定する．するとファイル一覧にいろいろなアイコンファイルが表示されるので好きなアイコンを選択して，OKボタンをマウスでクリックすれば，Helloプログラムにアイコンが付加される．

実際にface02.icoを選択してOK

LIST.1

```
Sub Command1_Click()....Command1というボタンがマウスで押された
    form2.Hide..........form2というウィンドウを見えなくする
End Sub.................処理の終了
```

LIST.2

```
Sub aboutitem_Click()...aboutというメニューがマウスで押された
    form2.Show..........form2というウィンドウを表示する
End Sub.................処理の終了
```

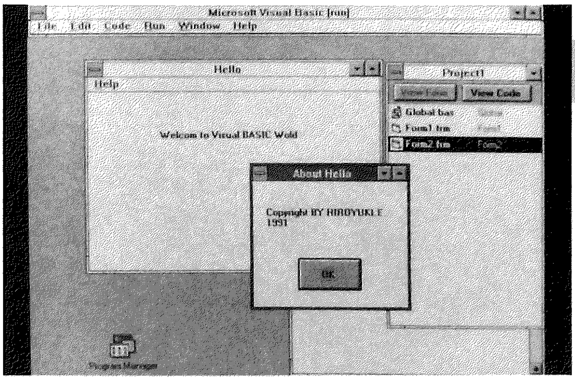


写真12
ダイアログボックスの表示

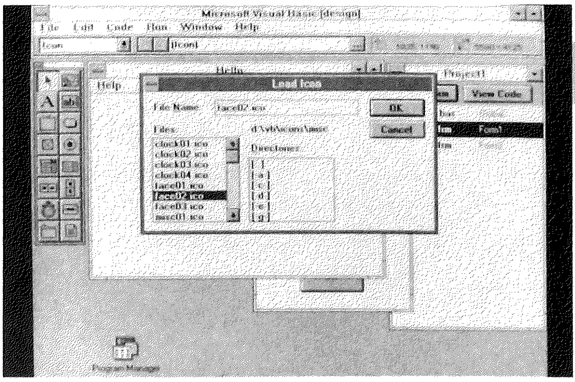


写真13
アイコンの選択

Windows ソフトウェア一覧
びっくす 2.0J，88,000円，プレゼンテーション用グラフィックソフト，アイネス

を実行してみる．RUNで，再度Helloプログラムを実行して最小化してみよう(写真14)．ニコちゃんマークのアイコンが表示されるだろう．

独自のアイコンは，付属しているアイコンエディタを用いて作成することができる．ほかにも標準で400個近いアイコンがついているので，その中から好きなものを選んで使うこともできる．

その他

プログラミングにおける機能を一部紹介したが，VisualBASICにはまだまだ多くの機能がある．DDEやWindows APIの使用なども，使い方さえ覚えてしまえば非常に簡単に行える．

VisualBASICには，電卓，カードファイル，アイコンエディタといったサンプルプログラムが付いているので，これらを見てDDEやWindows APIの使用方法などを勉強することができる(写真15)．また，電卓プログラムは3つのサンプルの中では比較的簡単なものなので，これを関数電卓に拡張してみるのもおもしろいだろう．

Visual BASICの問題点

VisualBASICは非常によくできた製品であるが，問題がないわけではない．サンプルプログラムのアイコンエディタを使ってみればわかると思うが，グラフィックの処理が非常に遅く感じる．マウスの速度にグラフィックの描画が追いつかないのである．できれば，もう少し早くなってほしいものだ．

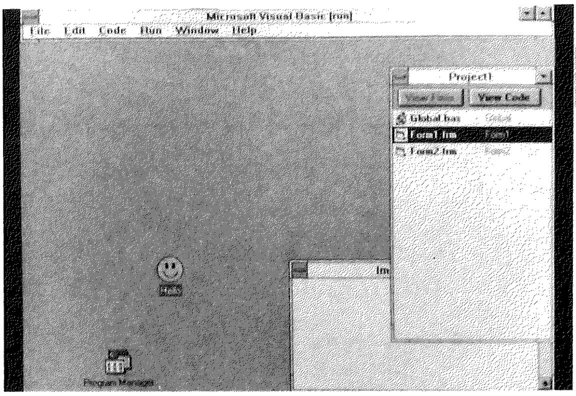


写真14
ニコちゃんマークのアイコンが表示される

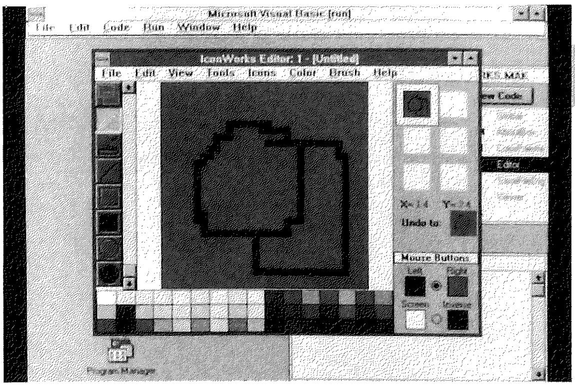


写真15
サンプルプログラム(アイコンエディタ)

ウィンドウズプログラミング普及の鍵に……

スクリーンセーバーやアイコンエディタ，テキストエディタなど，通常MS-C+SDKで作成できることは，ほとんどすべてVisualBASICでも可能である．また，DLLによってWindowsのAPIを使ったプログラミングもできる．簡単なデモ版などを早急に作らなければならない場合などには，まさにうってつけである．速度が遅くなることを了解した上でなら，そのままでも十分使用に耐えるプログラムを作成することが可能だ．

また，VisualBASICはランタイム型でプログラムを実行するため，ユーザーが作成したプログラムモジュール自体は，MS-C+SDKで作成されたものよりも小さくなる．このランタイムモジュールはライセンスフリーのため，VisualBASICで作成したアプリケーションと一緒に発売することができる．

さらに，BBSなどのネットワーク上での流通も問題なく行えるため，米国のCompuServeや国内の草の根BBS，大手パソコンネットでもランタイムモジュールの配布が行われている．このことから，今後VisualBASICを使ったオンラインソフトウェアが続々と出てくることはまちがいないだろう．すでに米国のCompuServeなどでは，VisualBASICで作成されたオンラインソフトが数多く出はじめている．

VisualBASICの日本語版が出れば，日本におけるWindowsとウィンドウズプログラミングの普及に大いに役立つと思うので，できるだけ早期に日本語化されることを望みたい．

ウィンドウ環境を生み出した GUIの元祖 Smalltalk/V Windows 1.1

編集部

PROGRAMMING Windows

Smalltalkは，けっして新しいプログラミング言語ではない．米ゼロックス社のパロアルト研究所で，アラン・ケイ（Alan Kay）らによって開発されたのが1972年というから，すでに10年近い歴史をもっていることになる．そのビットマップディスプレイ，オーバーラッピングウィンドウ，マウスオペレーションといった環境のすばらしさは高い評価を受けた．しかし，その環境を実現するには，当時のパーソナルコンピュータはあまりに貧弱すぎた．Smalltalk自体は，広く普及することはなかったが，その環境であるGUI(グラフィカルユーザーインターフェース)は，後のLisaやMacintoshに，さらにWindowsへと引き継がれたのである．このような経緯からしても，SmalltalkはWindowsプログラミングにもっとも適したプログラミング言語といえるだろう．

インストール

Smalltalk/V Windows（以下STVW）のインストールは，ほかのWindowsアプリケーション同様簡単なものである．Windowsのファイルマネージャからインストールプログラムを実行し，あとはインストールプログラムの指示に従ってSTVWをインストールするディレクトリを指定したり，ディスクを入れ換えていけばよい．

Ver.1.0からVer.1.1へのバージョンアップで変更されたおもな内容は，

①いくつかのクラス，メソッドが追加・変更された．
②アイコンエディタが付属した．
③ビットマップが変更された．

といった点である．これら以外にも細かな修正が施されている．

プログラミング環境

STVWでは，プログラミングのすべての作業をWindows内で行うことができる．もともとWindowsのようなビットマップ表示とオーバーラッピングウィンドウを備え，マウスを用いて操作する環境は，米ゼロックス社のパロアルト研究所で開発されたSmalltalkシステムからきているのだから，STVWのすべての操作がWindows内で行えるのは当然のことといえる．

また，STVWの基本的な操作はWindowsの操作方法に準拠しているが，ウィンドウ内のペインの上にマウスカーソルをおいてマウスの右ボタンをクリックすると，そのペインに合ったポップアップメニューが表れるという，Smalltalk特有のインターフェースももっている．いちいちメニューバーまでカーソルを移動してプルダウンメニューを表示させるよりも，ポップアップメニューを表示させるほうが早い．操作になれてきたら，ポップアップメニューを利用するとよいだろう．

STVWでは，数多くのウィンドウを利用してプログラミングをすることになる．ここで，それらのウィンドウについて簡単に説明する．

Transcript Window

STVW起動したときに表示されるのが，Transcript Windowである．このウィンドウは，STVWを使用しているときには常に表示され，閉じることはできない（アイコンにすることはできる）．

Transcript Windowは，テキストエディタとして用いるほか，選択したテキストをプログラムとして実行する機能を備えているので，簡単なプログラムのテストや変数の内容を調べるのに利用する．

Class Hierarchy Browser

このClass Hierarchy Browserが，STVWプログラミングの中心となるウィンドウだ（写真1）．クラス，サブクラスの階層構造が表示され，これらのソースを参照したり，変更したりすることができる．もちろん，新しいクラスやメソッドを定義することも可能だ．

Sender and Implementor Browsers

このブラウザは，あるクラス中のメソッドから呼び出されているメソッドを表示するなど，メソッド間のクロスレファレンスを調べるのに利用される．

Inspector

Inspectorは，オブジェクトの内容を調べたり，変更したりするのに用いられる．デバッグに利用すると便利である．

Walkback

Walkbackウインドウは，プログラムを実行したときにエラーが生じると自動的に表示される．表示される内容は，エラーをおこしたときのプログラムの状態である．このあと，デバッガを起動するなりなんなりしてバグを修正する．

Debugger

その名のとおりデバッガである．このデバッガは，Walkbackの中から呼び出される．

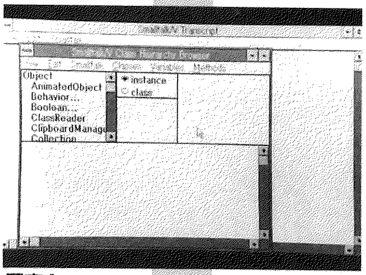


写真1　Class Hierarchy Browser

Smalltalk プログラミングの考え方

Smalltalkによるプログラミングは，Cなどを使用したプログラミングとは根本的に異なったものである．Cなど，通常のプログラミングになれた人は，頭を切り替えて取り組む必要がある．

Smalltalkプログラミングにおける3つのポイントについて述べる．

モードレス

一般のプログラミングには，

①エディタを起動してプログラムをエディットする
②コンパイラを起動してコンパイルする
③リンカを起動してリンクする
④プログラムを実行する
⑤バグがあればデバッガを起動してデバッグする
⑥もう一度②にもどる

というように，複数のモードが存在している．しかし，Smalltalkでは，エディタ，コンパイラ，リンカ，デバッガ，さらにはOSまでが完全に統合されているため，このようなモードというものが存在しない．

プログラマは，プログラムをすべて書き終えるのを待つことなく，いつでもプログラムをコンパイルしたり，デバッグしたり，実行することができる．モードレスになっているので，部分的に試行錯誤を繰り返しながら，プログラムを完成させていくことが可能である．

ブラウジング

ブラウジング(Browsing)とは，拾い読みのことだ．先にも述べたように，STVWのプログラミングの中心になるのは，Class Hierarchy Browserと呼ばれるブラウザである．

つまり，Smalltalkのプログラミングとは，Smalltalkがもつクラスライブラリ，またはシステムそのものを拾い読み(ブラウズ)して，それらに変更を加えたり，カット&ペーストで自分のプログラムに取り込んだりすることなのである．

MVCモデル

MVCモデルとは，プログラムを大きくモデル(Model)，ビュー(View)，コントローラ(Controller)の3つのオブジェクトに分けて構成する考え方のことである．

モデルとは，シミュレーションの対象とでもいうべき動作モデルやデータモデルなど，具体例を思い浮かべることができるものをいう．

ビューとは，このモデルをどのような角度からとらえるかという視点のことである．

最後のコントローラは，ビューを通してモデルに変化を起こさせるような機能を果たすものである．

マルチウィンドウを使用したアプリケーションプログラムを，MVCモデルで表現すると，モデルは操作されるデータとその動作，つまりアプリケーションの機能を実現するオブジェクトであり，ビューはデータを表示するウィンドウのオブジェクト，コントローラはアプリケーションに対する制御を行うオブジェクトということになる．

このMVCモデルは，ワークステーションで動作するオリジナルのSmalltalkに採用されたものだが，パーソナルコンピュータ向けに開発されたSmalltalk/Vシリーズにも，そのまま受け継がれている．Smalltalkでアプリケーションを作成する場合，必ず，このMVCモデルに準拠した形で，オブジェクトを構成していくことを覚えておいて欲しい．

実際のプログラミング

STVWのプログラミングでもっとも重要なことは，“re-invent the wheel (車輪を2度発明する)” 愚を犯さないことだ．まず，自分が作ろうとしているプログラムにもっともあっていると思われるクラスを捜し出し，この下位クラス(オブジェクト)としてアプリケーションを定義する．次に，上位のクラスから継承したメソッドに変更を加えたり，新たなメソッドを付け加えることでアプリケーションを完成させていく．

このように既存のものに，変更・追加をしてプログラミングしていくことを，インクリメンタルプログラミング(差分プログラミング)と呼んでいる．Cのプログラマがよくやるような，コンパイラに付属する標準ライブラリすら使用せず，1からすべてのコードを書き起こしていくというのは，Smalltalkの世界では無縁のものである．

Windows ソフトウェア一覧
美筆 ARTS&LETTERS V3.0　248,000円，グラフィック応用プレゼンテーションツール，ノックスインターナショナル

次に，Class Hierarchy Browserを用いたプログラミングを具体的に示す．

クラスの追加

まず，Transcript WindowのFileメニューからBrowse Classesを選んで，Class Hierarchy Browserを立ち上げる．次にClass Hierarchy Browserの左上に表示されているクラス名の中から，ViewManagerを選んでマウスでクリックし，さらにClassesメニューからAdd Subclass...を選択する．すると，サブクラス追加用のダイアログボックスがポップアップするので，ここで追加するサブクラス名(Test)を入力する．

STVWでアプリケーションを作成するときには，一般的にViewManagerの下にサブクラスを追加する．

メソッドの追加

次に，この新しいクラスにメソッドを追加してみる．

Class Hierarchy BrowserのMethodsメニューからNew Methodを選ぶと，下のテキストペインにメソッドのテンプレートが表示される．このテンプレートを元に，リスト1のようなopenメソッドを定義する．メソッドを記述し終ったなら，FileメニューからSaveを選択してメソッドを登録する(写真2)．

グローバル変数の登録

さて，クラスとメソッドの追加が終ったら，次にTranscriptウィンドウにもどってグローバル変数を登録する．Transcriptウィンドウのテキストペインに，「RunTest := Test new」と記述して，これをShow itする．するとダイアログボックスがポップアップして，RunTestをグローバル変数として登録するかどうかと聞いてくるので，Yesボタンをマウスでクリックして登録する．

プログラムの実行

以上で，簡単ではあるがプログラムが完成したので実行してみよう．

Transcriptウィンドウに「RunTest open」と記述して，これをShow itする．すると，写真3のように，プログラミングしたとおりのウィンドウが起動する．

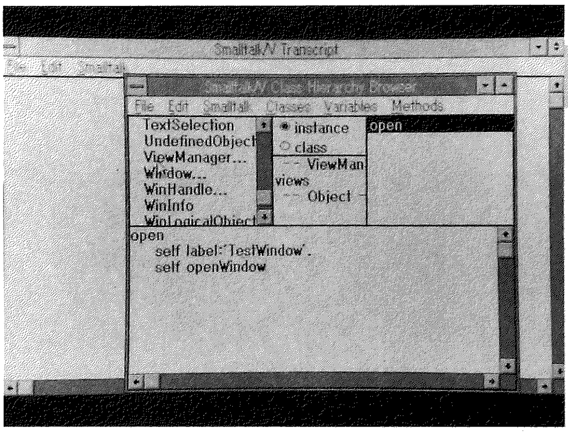


写真2
メソッドを追加する

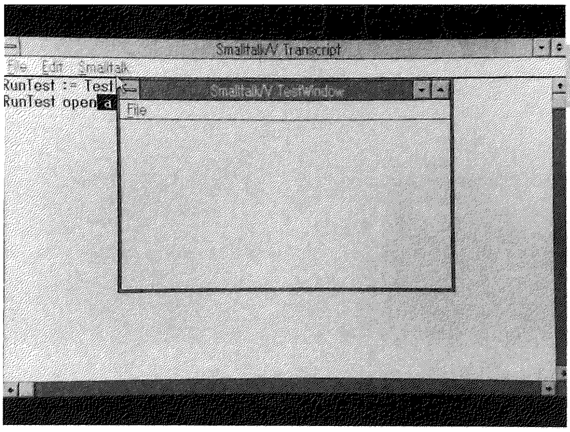


写真3
Testを実行する

LIST.1
openメソッド

```
open
        self label: 'TestWindow'.
        self openWindow
```

特集 Windows プログラムの作り方

クラスライブラリ，リソース作成，SQL処理など

Windowsプログラム開発支援ツール

編集部

複雑なWindowsプログラムの作成を少しでも容易にするために，各社からさまざまなツールが発売されている．ここでは，それらの製品の中からいくつかをピックアップして紹介する．

Actor 3.1 Professional

リソース作成ツールとクラスライブラリを備える

Actorは，Smalltalk/V Windowsとよく似た統合開発環境をもつオブジェクト指向プログラミング言語である．しかし，Smalltalk/V Windowsがもともとワークステーション上の処理系であったのに対し，Actorは1987年に発売されたバージョン1.0以来，純粋にWindows専用の処理系として発達してきた．

本誌2月号の特集「Windowsの時代である！」では，Actor 3.0を紹介したが，その後3.1にバージョンアップされ，さらにActor 3.1 ProfessionalとActor 3.1の2つのプロダクツに分けて発売されるようになった．Professionalには，Whitewater Resource Toolkit, ObjectGraphicsという2つの開発ツールが含まれている．

ここでは，Professionalに含まれているWhitewater Resource ToolkitとObjectGraphicsについて紹介する．

Whitewater Resource Toolkit

Windowsプログラムには，ダイアログボックス，ビットマップ，カーソル，アイコン，メニューなど，リソースと呼ばれる部品が存在している．MS-C+SDKでは，SDKペイントやダイアログエディタなどのSDK付属のツールを使用したり，普通のソースのようにテキストエディタで入力したりして，これらのリソースを作成することになる．Whitewater Resource Toolkit(以下WRT)は，これらのリソースを効率よく作成・編集するためのツールである．

WRTは，実行形（exeファイル），リソースファイル（resファイル），ビットマップファイル（bmpファイル），カーソルファイル（curファイル），アイコンファイル(icoファイル)を直接編集・作成することができる．また，これらをSDKのリソースコンパイラと互換性のあるテキストファイルとしてセーブすることも可能である．

WRTの操作方法は，非常に簡単である．WRTを立ち上げ，編集・作成したいリソース，またはリソースファイルを選び，あとはそれぞれのツールを用いてグラフィカルにリソースを編集・作成すればよいだけである．

また，WRTは直接プログラムの実行形を編集することができるので，既存のWindowsプログラムのアイコンやカーソルなどを，ユーザーの好みに合わせて変更することもできる．

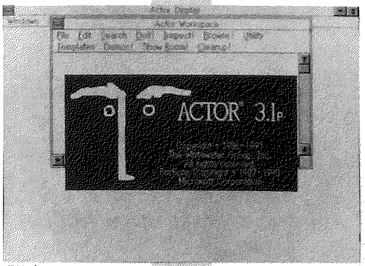


写真1　Actor 3.1 Professional

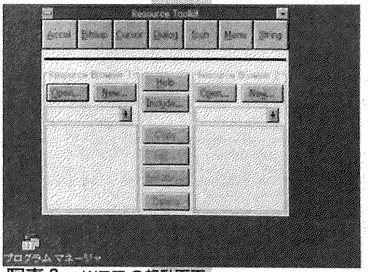


写真2　WRTの起動画面

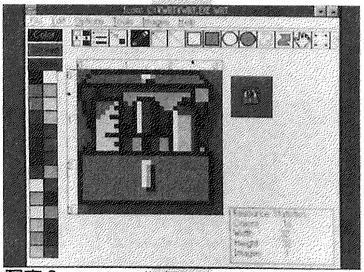


写真3　アイコン作成画面

ワンライン NEWS
日本IBMは滋賀県の野洲工場で稼働中のLSI製造行程管理システムの販売を開始．半導体メーカーを対象とし，同社が推進するソリューション型ビジネスの一環にする．

ObjectGraphics

ObjectGraphics（以下OG）は，Actorに組み込んで使用するクラスライブラリ集である．OGには，グラフィックスに関する30を越えるクラスが含まれている．これらのクラスは，次の3つのカテゴリに分けられる．

【Rendering tools】

このカテゴリには，グラフィックオブジェクトの描画をコントロールするためのクラスが含まれている．これらの中には，Pen，Brush，TextPen，GraphSpaceなどがある．

グラフィックオブジェクトは，これらを所有して，自分自身をビットマップやプリンタに描き込む．

【Platform filters】

このカテゴリのクラスは，グラフィックコードの移植性を高めるために使用される．このカテゴリに含まれるクラスには，Port，PhysicalPort，Bitmap，Colorなどがある．

これらのクラスを使用すると，ウィンドウ，ビットマップ，プリンタなどに対してまったく同じ方法でグラフィックスを描き込むことができるようになる．

【Graphics】

このカテゴリのクラスは，実際に描かれるグラフィックスそのものである．このカテゴリに含まれるクラスには，Icon，Bitmapなどがある．

OGには，実際にOGを使用して作られたObjectDrawというお絵描きソフトがサンプルとして付属している．また，マニュアルにはこのObjectDrawを作成していく過程が，こと細かに解説されているので，OGの具体的な利用法を学ぶよいチュートリアルになるだろう．

●製品名：Actor 3.1 Professional
●価格：Actor 3.1 Professional 188,000円／Actor 3.1（日本では未発売）／Whitewater Resource Toolkit 58,000円／ObjectGraphics 58,000円
●問い合わせ先：（有）データポット TEL 052-798-4154

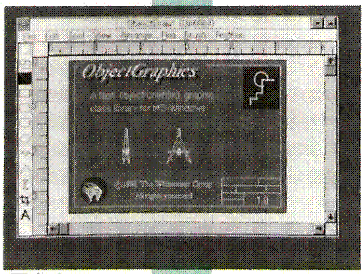
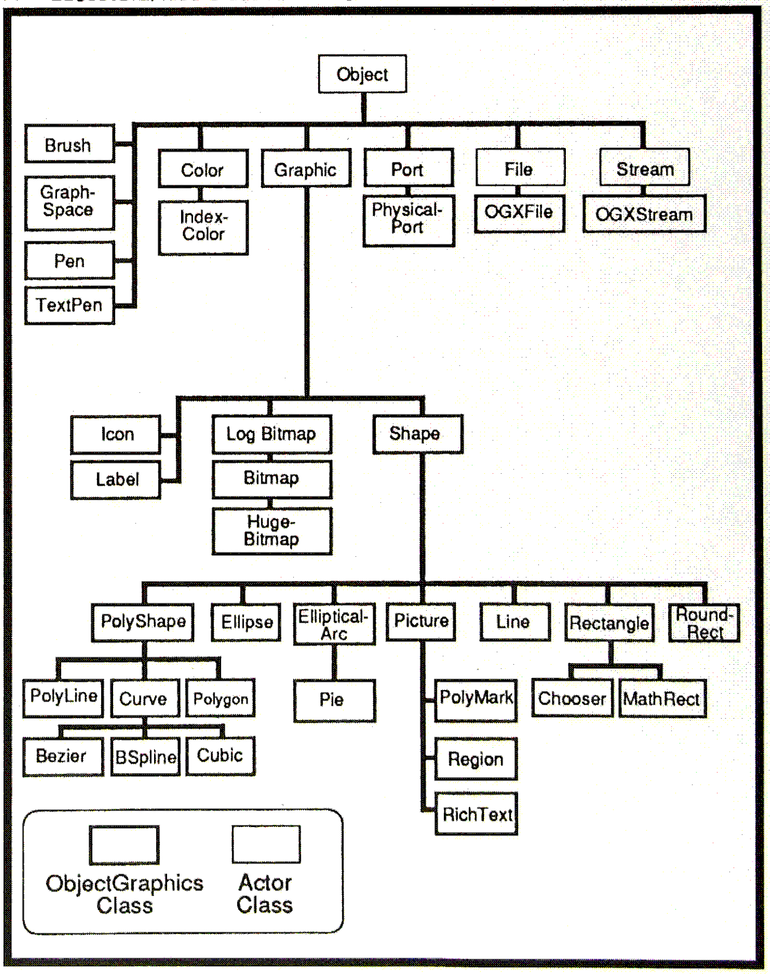


写真4　ObjectDraw

図1　ObjectGraphics Class Hierarchy



C++/Views

Smalltalk ライクなブラウザを備えたC++クラスライブラリ

C++/Viewsは，60を越えるクラスを含むC++用のクラスライブラリであると同時に，Windows上で動作するSmalltalkライクなブラウザを含むプログラミングツールである．

C++/Viewsにおけるプログラミングの中心となるのは，SmalltalkやActorと同様にブラウザである．このブラウザは，アプリケーション開発の

ために次の4つのファイルを利用する.

【＊.envファイル】

これは，アプリケーションを構成するすべてのクラスに関する情報が収められているファイルである.

【＊.mファイル】

アプリケーションを作るために利用されるメイクファイルのテンプレート.

【＊.lファイル】

リンカ(LINKまたはLINK4)のための定義ファイルを作成するためのテンプレート.

【＊.dファイル】

モジュール定義ファイル（＊.def)を作るためのテンプレート.

C++/Viewsで新しいアプリケーションを作成するときには，まずブラウザのApplicationメニューからOpenを選び，all.envというファイルを読み込んで，さらにApplicationメニューからSave As...を選択し，all.envをリネームして使用する．また，C++/Viewsで使用するコンパイラなどは，cb.cfgファイル中に記述しておく.

C++/Viewsは，C++ Version 2.0に準拠したC++コンパイラおよびC++プリプロセッサに対応している．マニュアルには，対応コンパイラとしてZortech C++ 2.0，プリプロセッサとしてGlockenspielがあげられている．当然のことだが，プリプロセッサを使用した場合にはMS-CなどのWindowsに対応したCコンパイラが必要になる.

C++/Viewsが対応しているメモリモデルはラージモデルであり，コンパイラ，プリプロセッサ以外にマイクロソフト社が発売しているWindows SDKが必要である.

C++/Viewsには，ブラウザ以外にも次のようなユーティリティが付属している.

【Mkarc】

Mkarcはいわゆるアーカイバとして働くほかに，アーカイブされたクラスのソースファイルやヘッダファイルからドキュメントを作成したり，アプリケーション環境ファイル(.envファイル)をアップデートしたりする機能をもっている.

【Mkdoc】

Mkdocは，ブラウザやMkarcによって作られたドキュメントファイルからclasses.outという名前のワープロ（デフォルトはMS Word）用のファイルを作成する.

【Mkdlg】

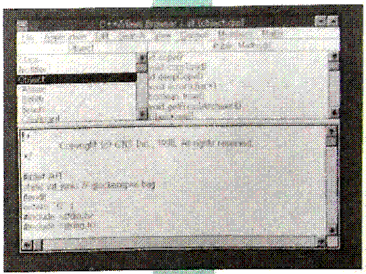
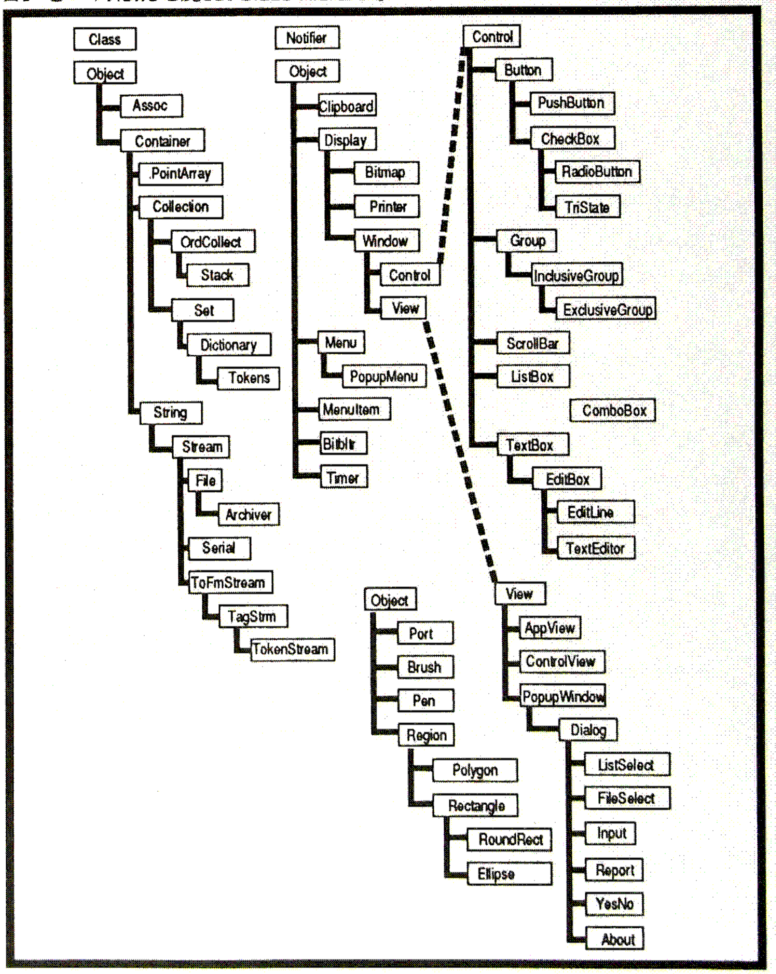


写真5　C++/Viewsのブラウザ

Mkdlgは，SDKに付属しているダイアログエディタの出力したファイルをC++に適した新しいダイアログサブクラスに変更するツールである.

●製品名：C++/Views
●価格：154,500円
●問い合わせ先：システム・ワン（株）
TEL 03-3663-4213

図2　C++/Views Object Class Hierarchy



ワンライン NEWS
キヤノンが発売したAXパソコン用MS-Windows 3.0は米ノベル社のネットワークOSに対応している．国内で同OSに対応したWindows 3.0ははじめて.

SQLWindows

RDB の標準言語，SQL を使ったデータベース処理

SQLWindowsは，Windows上のビジュアルプログラミングを実現するアプリケーション開発ツールである．既存のプログラミング言語を使わず，画面上でマウスを使って画面設計を行うだけで，自動的にプログラムコードが生成される点が大きな特徴である．

インストールを終了すると，「SQLWindows」グループが作られる（写真6）．その中からSQLWindowsを実行すると，画面に左右2つのウィンドウが表示される．左側のウィンドウのメニューからコマンドを選んで，右側のウィンドウにデータ入力ボックスや，各種のボタンなどを配置して画面設計を行うと，左側のウィンドウにはそれに対応する命令が自動的に生成され，記述されていく（写真7）．

画面上に定義したそれぞれのオブジェクトには，データのタイプ，最大長，表示形式などのさまざまなオプションが指定できる．実行処理部分のコーディングについては，画面上のオブジェクトに付随するイベントに対するアクションとしてプログラムする．20以上のイベント検知機能，200以上のビルトイン関数，定義済みのシステム関数が利用できるほか，ユーザーが独自にC言語で書いた外部関数を利用するための機能もあらかじめ備えられている．

SQLWindowsがもっとも得意とするアプリケーションは，SQL言語を使ったデータベース処理プログラムである．SQLとは，Structured Query Language（構造化問い合わせ言語）の頭文字を取ったもので，リレーショナルデータベースにおけるデータ定義，データの操作，およびこれを操作する機能をもつ．米国IBM社が提唱して以来，各社からSQL対応製品が発表され，86年にはANSIでも規格として認定されたリレーショナルデータベースの世界での標準言語である．

SQLWindows開発システムには，SQLBaseシングルユーザー版がバンドルされているため，このシステムを購入するだけで，スタンドアローンのデータベースを構築し，データの追加・更新・検索などを行うアプリケーションを作ることができる．別売のSQLBase Server（MS-DOS版，OS/2版）と組み合わせれば，クライアントサーバー方式のデータベースシステムを構築することもできる．その場合には，クライアントマシン側でユーザーインターフェースを担当し，サーバーマシン側でデータベースの検索・更新・削除などの処理を行う．現在MS-Networks，NetWare，LAN MANAGERなどの各種LAN OSをサポートしている．また，SQL Networkファミリーを利用すれば，DB2やOracleなどメインフレームやミニコン上のデータベースへも接続することができるため，大規模な企業情報システムのユーザーインターフェースの作成に利用することも可能である．

SQLWindows開発システムには，SQLWindowsアプリケーション開発システム，SQLBaseシングルユーザーデータベースエンジン，SQLTalk/Windows対話型データベースマネージャ，SQLTalk（キャラクタモード），Express Windowsアプリケーションジェネレータ，Report Windowsグラフィカルレポートライター，SQL/API Cアプリケーションプログラムインターフェースが含まれる．

なお，このアプリケーション開発環境を用いて開発されたアプリケーションプログラムをほかのマシンで利用するためには，SQLWindows環境でコンパイルして，中間実行モジュールを生成するだけでなく，別売のランタイム製品を購入する必要がある．

また，92年春には日本語文字列に対応したバージョンの出荷が予定されている．

対応機種はIBM PC/AT，PS/2とその互換機，日本IBM PS/55シリーズ，各社AXパソコン，NEC PC-9801シリーズで80286以上のCPUを搭載した機種．4MB以上のRAM，20MBのハードディスクが必要で，Windows3.0のスタンダードモード，またはエンハンストモードで使用する．

●製品名：SQLWindows開発システム Ver.2.0
●価格：298,000円
●問い合わせ先：三菱商事（株）技術第一部ネットワーク取引開発タスクフォース　TEL 03-3210-7529

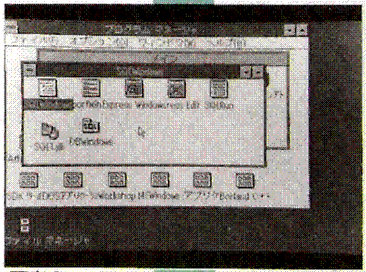


写真6　SQLWindowsグループ

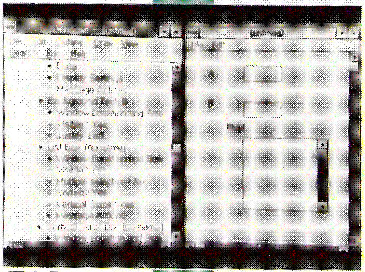


写真7　SQLWindows実行中

プログラム遊工房

谷口典生

十乃巻 「SNAKE」ゲーム

今回は前回に続いてアクションゲームを作ってみました．ゲームとしては未完成といった感じのプログラムですが，蛇のようなキャラクタの動きを楽しんでもらえればよいと思います．ワンポイントテクニックでは，VSYNC割り込みを使ったゲームスピードの調整方法について解説しています．

リスト1

```
/*
 * snake.c              「SNAKE」ゲーム
 */

#include <time.h>
#include <dos.h>

#ifndef __TURBOC__      /* Turbo C → MS-C(Quick C) 関数名変換 */
#define enable()        _enable()
#define inportb(x)      inp(x)
#define outportb(x,y)   outp(x,y)
#define setvect(x,y)    _dos_setvect(x,y)
#define getvect(x)      _dos_getvect(x)
#endif

#ifdef __TURBOC__
/* Turbo C 用グラフィック初期化/復帰ルーチン */
#include <graphics.h>
void graphinit()                        /* グラフィックモード設定 */
{
        int graphdriver, graphmode;

        graphdriver = DETECT;           /* 自動検出 */
        graphmode = PC98C8;             /* 8色モード */
        initgraph(&graphdriver, &graphmode, "");
}
void graphdeinit()                      /* グラフィックモード復帰 */
{
        restorecrtmode();               /* 画面モードを元に戻す */
        printf("¥x1b*");                /* 画面クリア */
}
#else
/* MS-C, Quick C 用グラフィック初期化/復帰ルーチン */
#include <graph.h>
#define BLACK                   0
#define GREEN                   2
#define CYAN                    3
#define MAGENTA                 5
#define WHITE                   7
#define setcolor                _setcolor
#define lineto                  _lineto
#define moveto                  _moveto
#define rectangle(l,t,r,b)      _rectangle(_GBORDER,l,t,r,b)
#define circle(x,y,r)   _ellipse(_GBORDER,(x)-(r),(y)-(r),(x)+(r),(y)+(r))
void graphinit()                        /* グラフィックモード設定 */
{
        _setvideomode(_98RESSCOLOR);    /* グラフィック8色+テキストモード */
        printf("¥x1b[>5h");             /* カーソル消去 */
}
void graphdeinit()                      /* グラフィックモード復帰 */
{
        _setvideomode(_DEFAULTMODE);    /* デフォルト(テキスト80行)モード */
        printf("¥x1b[>5l¥x1b[>1l");     /* カーソル表示、最下行復帰 */
}
#endif

#define MAX_LENGTH      100             /* 蛇の長さの最大値 */
#define MAX_FOODS       10              /* 餌の最大数 */
#define MAX_ENEMYS      10              /* 敵の最大数 */
#define dist(x,y)       ((long)(x)*(x)+(long)(y)*(y))

struct object_struct {                          /* 敵と餌のXY座標、方向 */
        int x, y;
        int dir;
} enemy[MAX_ENEMYS], food[MAX_FOODS];

int x_max, x_min, y_max, y_min;                 /* 蛇や敵の動く範囲 */
int d2x[36], d2y[36];                           /* 角度→XY値のテーブル */
int head_x, head_y, head_dir;                   /* 蛇の頭の座標、方向 */
int body_x[MAX_LENGTH], body_y[MAX_LENGTH];     /* 蛇の胴体の座標 */
int length;                                     /* 蛇の胴体の長さ */
int gameoverflag=0;                             /* ゲーム終了フラグ */
int enemys, foods;                              /* 敵の数、餌の数 */
long counter=0;                                 /* VSYNC割り込みカウンタ */
void (interrupt far *oldvsync)();               /* 元のVSYNC割り込みルーチン */

void interrupt far vsync()              /* VSYNC割り込みルーチン */
{
        counter++;                      /* カウンタ値を増やす */
        oldvsync();                     /* 元のVSYNC割り込みを呼び出す */
        outportb(0x64, 0);              /* VSYNC割り込みをリセット */
}

void set_vsync()                /* 割り込みルーチンの設定 */
{
        outportb(2, inportb(2) | 4);    /* VSYNC割り込みをマスク */
        oldvsync = getvect(0x0a);       /* 元の割り込みベクタを保存 */
        setvect(0x0a, vsync);           /* vsync() をベクタに設定 */
        outportb(2, inportb(2) & ~4);   /* マスクのクリア */
        outportb(0x64, 0);              /* VSYNC割り込みのリセット */
```

プログラムの使い方

このプログラムは，PC-9801シリーズ用で，Turbo C, QuickCおよびMS-Cに対応しています．リスト1を打ち込み，コンパイルして実行ファイルを作ってください．(MS-Cでコンパイルするときは/Olオプションをつけないでコンパイルしてください)．

今回のプログラムではグラフィック関数を使っているので，MS-CやQuickCでコンパイルするときにはグラフィックライブラリを指定します．MS-Cなら「CL snake.c /link graphics.lib」などとします．Turbo Cでは実行時に，BGIファイルが必要となるので注意してください．

プログラムの実行は，

SNAKE 〈敵の数〉 〈餌の数〉

と，2つの数字（それぞれ0から10の範囲）を引数に指定します．ゲームの速度はCPU速度に依存しないようになっていますが，ある程度数が大きくなると機種によっては遅くなるので注意してください．

```
        enable();                       /* 割り込み許可フラグをセット */
}

void restore_vsync()            /* 元の割り込みルーチンの復帰 */
{
        outportb(2, inportb(2) | 4);    /* VSYNC割り込みをマスク */
        setvect(0x0a, oldvsync);        /* 割り込みベクタを元に戻す */
}

void generate(struct object_struct *obj)        /* 餌や敵を生成する */
{
        obj->x = x_min + rand() % (x_max-x_min+1);
        obj->y = y_min + rand() % (y_max-y_min+1);
        obj->dir = rand() % 36;
}

void init()     /* '角度→XY値' テーブルの設定、イニシャライズ */
{
        int i;
        static int sind[10]={0, 173, 342, 499, 642, 766, 866, 939, 984, 1000};

        for(i=0; i<36; i++) {
                if(i<=9) {                      /* 0～90度 */
                        d2x[i] = sind[9-i];
                        d2y[i] = sind[i];
                } else if(i<=18) {              /* 100～180度 */
                        d2x[i] = -sind[i-9];
                        d2y[i] = sind[18-i];
                } else if(i<=27) {              /* 190～270度 */
                        d2x[i] = -sind[27-i];
                        d2y[i] = -sind[i-18];
                } else {                        /* 280～350度 */
                        d2x[i] = sind[i-27];
                        d2y[i] = -sind[36-i];
                }
        }
        x_max = 628;    x_min = 12;             /* 物体(の中心)が動く範囲 */
        y_max = 388;    y_min = 12;
        head_x = 320;   head_y = 200;           /* 蛇の頭の初期位置 */
        head_dir = 0;                           /* 蛇の動く方向 */
        length = 10;                            /* 蛇の胴体の長さ */
        for(i=0; i<MAX_LENGTH; i++) {
                body_x[i] = head_x;
                body_y[i] = head_y;
        }
        setcolor(WHITE);                        /* フィールド枠を描く */
        rectangle(0, 0, 639, 399);
        for(i=0; i<enemys; i++)                 /* 敵の初期データ設定 */
                generate(&enemy[i]);
        for(i=0; i<foods; i++)                  /* 餌の初期データ設定 */
                generate(&food[i]);
}

void rebound(int *x, int *y, int *dir)          /* 壁に当ったら跳ね返る */
{
        if(*x > x_max) {                /* 右の壁に当ったか */
                *x = x_max*2 - *x;      /* X座標を壁に対して対称な位置へ */
                *dir = 18 - *dir;       /* 反射した角度にする */
        } else if(*x < x_min) {         /* 左の壁に当ったか */
                *x = x_min*2 - *x;
                *dir = 18 - *dir;
        }
        if(*y > y_max) {                /* 下の壁に当ったか */
                *y = y_max*2 - *y;
                *dir = -*dir;
        } else if(*y < y_min) {         /* 上の壁に当ったか */
                *y = y_min*2 - *y;
                *dir = -*dir;
        }
        *dir = (*dir+36) % 36;          /* 0～35 の範囲に修正 */
}

void movesnake()                /* 蛇を動かす */
{
        unsigned char far *kp=(unsigned char far *)0x00000531L;
        int i;

        if(*kp&8)                       /* '←' キーが押されているか */
                head_dir++;             /* 角度を正方向へ増やす */
        if(*kp&16)                      /* '→' キーが押されているか */
                head_dir--;             /* 角度を負方向へ増やす */
        head_dir = (head_dir+36) % 36;
        head_x += d2x[head_dir]/100;    /* 角度に応じてX座標を増分 */
        head_y -= d2y[head_dir]/100;    /*                Y座標         */
        rebound(&head_x, &head_y, &head_dir);
        setcolor(BLACK);                /* 最後尾を消す */
        circle(body_x[length-1], body_y[length-1], 10);
        for(i=MAX_LENGTH-1; i>=1; i--) {        /* 胴体の位置データをシフト */
                body_x[i] = body_x[i-1];
                body_y[i] = body_y[i-1];
        }
        body_x[0] = head_x;             /* 頭の座標を代入 */
        body_y[0] = head_y;
        setcolor(GREEN);                /* 新しい位置に頭を描く */
        circle(head_x, head_y, 10);
}
```

ゲームの遊び方

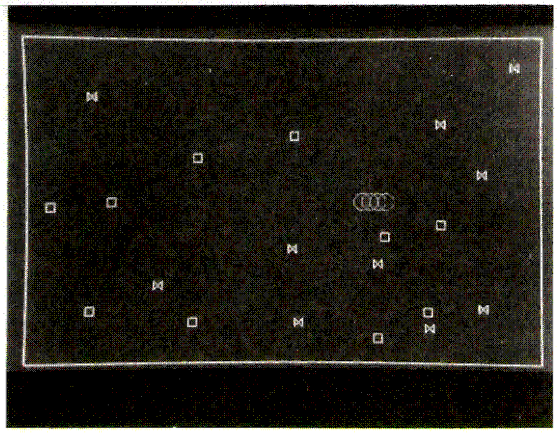
プログラムを実行すると，画面中央から右に向かって動き出す緑色の円が連なったキャラクタが蛇です．矢印キー（カーソルキー）の左右を押すと1歩ごとに進行方向に対して10度ずつ向きを変えることができます．壁にぶつかった場合は壁に対して反射します．

紫色の四角いキャラクタが敵で，壁に当たると反射し，ときどき方向を変えながら動いています．蛇の頭がこのキャラクタに当たると蛇の胴体が1つ短くなってしまいます．

水色の蝶型のキャラクタは餌で，敵と同じような動きをしていますが，蛇の頭がこのキャラクタに当たると蛇の胴体は1つ長くなります．

最初の蛇の胴体の長さは10ですが，敵にぶつかって長さが0になってしまうとゲームオーバーです．長さが0になるまでゲームは終了しないので，途中でやめたいときはESCキーを押してください．stopキーなどで無理に実行を中断すると割り込みベクタを元に戻さずに終了してしまうので暴走する危険があります．

```
187:
188: void moveenemy(int num)         /* 敵を動かす */
189: {
190:         setcolor(BLACK);                /* 前の位置を消去 */
191:         rectangle(enemy[num].x-5, enemy[num].y-5,
192:                 enemy[num].x+5, enemy[num].y+5);
193:         if(dist(enemy[num].x-head_x, enemy[num].y-head_y) < 450) {
194:                 generate(&enemy[num]);  /* 新しく生成(蛇の頭にぶつかった) */
195:                 setcolor(BLACK);        /* 蛇の胴体最後尾を消去 */
196:                 circle(body_x[length-1], body_y[length-1], 10);
197:                 if(length==1)           /* 蛇の長さが1ならゲームオーバー */
198:                         gameoverflag = 1;
199:                 else                    /* 2以上なら長さを1減らす */
200:                         length--;
201:                 return;
202:         }
203:         enemy[num].x += d2x[enemy[num].dir]/130;
204:         enemy[num].y -= d2y[enemy[num].dir]/130;
205:         rebound(&enemy[num].x, &enemy[num].y, &enemy[num].dir);
206:         if(enemy[num].x>head_x)         /* 蛇の頭の方向へ少し位置を修正する */
207:                 enemy[num].x--;
208:         else if(enemy[num].x<head_x)
209:                 enemy[num].x++;
210:         if(enemy[num].y>head_y)
211:                 enemy[num].y--;
212:         else if(enemy[num].y<head_y)
213:                 enemy[num].y++;
214:         setcolor(MAGENTA);              /* 新しい位置に表示 */
215:         rectangle(enemy[num].x-5, enemy[num].y-5,
216:                 enemy[num].x+5, enemy[num].y+5);
217:         if(rand()%25==0)                /* 1/25の確率で進行方向を変える */
218:                 enemy[num].dir = rand() % 36;
219: }
220:
221: void putfood(int color, int x, int y)           /* 餌の表示 */
222: {
223:         setcolor(color);
224:         moveto(x-5,y-5);        /* 蝶型(横向きの砂時計型)に表示 */
225:         lineto(x+5,y+5);
226:         lineto(x+5,y-5);
227:         lineto(x-5,y+5);
228:         lineto(x-5,y-5);
229: }
230:
231: void movefood(int num)          /* 餌を動かす */
232: {
233:         putfood(BLACK, food[num].x, food[num].y);       /* 消去 */
234:         if(dist(food[num].x-head_x, food[num].y-head_y) < 450) {
235:                 generate(&food[num]);   /* 新しく生成(蛇に食われた) */
236:                 if(length<MAX_LENGTH)   /* 蛇が最大長以内なら長さを1増やす */
237:                         length++;
238:                 return;
239:         }
240:         food[num].x += d2x[food[num].dir]/210;
241:         food[num].y -= d2y[food[num].dir]/210;
242:         rebound(&food[num].x, &food[num].y, &food[num].dir);
243:         putfood(CYAN, food[num].x, food[num].y);        /* 表示 */
244:         if(rand()%50==0)                /* 1/50の確率で進行方向を変える */
245:                 food[num].dir = rand() % 36;
246: }
247:
248: void game()                     /* メインループ */
249: {
250:         int i;
251:         long lastcount;
252:
253:         lastcount = counter;            /* VSYNCカウンター値を取得 */
254:         while(!gameoverflag) {          /* ゲームオーバーになるまでループ */
255:                 while(kbhit()) {
```



プログラムの説明

8-14行

MS-C（QuickC）の関数をTurbo Cの関数名に合わせるための定義.

16-55行

Turbo CとMS-C（QuickC）のグラフィック関数の違いを吸収するための定義.

vsync()
set_vsync()
restore_vsync()

VSYNC割り込みルーチン本体とその設定，復帰用ルーチン．ワンポイントテクニックの項を参照.

generate()

餌や敵の生成．ランダムな場所と方向を設定します.

init()

角度からXY値への変換テーブルを作成し，ゲームの初期設定をします.

112-126行

このプログラムでは角度を0から35の整数値で扱っていて，進行方向との対応は右向きを0として左回りに10度きざみの数値になっています．d2x［］とd2y［］はその角度値からcos, sinを1000倍した値への変換テーブルで，0度から90度までのsin値(の1000

```
256:                    if(getch()==0x1b)       /* ESCキーが押されたら戻る */
257:                            return;
258:                }
259:                movesnake();                /* 蛇を動かす */
260:                for(i=0; i<enemys; i++)
261:                        moveenemy(i);       /* 敵を動かす */
262:                for(i=0; i<foods; i++)
263:                        movefood(i);        /* 餌を動かす */
264:                while(counter-lastcount<3); /* 時間待ち */
265:                lastcount = counter;        /* VSYNCカウンター値を取得 */
266:        }
267: }
268:
269: int main(int ac, char *av[])
270: {
271:        time_t seedtime;
272:
273:        if(ac<3) {      /* パラメータが不足していたら使い方を表示して終了 */
274:                printf("Usage: snake <敵の数(0-10)> <餌の数(0-10)>¥n");
275:                exit(0);
276:        }
277:        enemys = atoi(av[1]);           /* 第1パラメータを敵の数に変換 */
278:        foods = atoi(av[2]);            /* 第2パラメータを餌の数に変換 */
279:        if(enemys<0 || enemys>MAX_ENEMYS || foods<0 || foods>MAX_FOODS) {
280:                printf("パラメータが不正です¥n");
281:                exit(1);
282:        }
283:
284:        time(&seedtime);
285:        srand((unsigned int)seedtime);          /* ランダマイズ処理 */
286:        graphinit();                            /* 画面モードの初期化 */
287:        init();                                 /* ゲームの初期化 */
288:        set_vsync();                            /* VSYNC割り込みをセット */
289:        game();                                 /* メインルーチン */
290:        restore_vsync();                        /* VSYNC割り込みを復帰 */
291:        getch();
292:        graphdeinit();                          /* 画面モード復帰 */
293: }
294:
```

倍）のテーブル（110行目のsind []）から作成しています．sinなどのライブラリ関数を使用してもよいのですが，この方法だとよけいなライブラリをリンクしなくてすむのでプログラムサイズの節約になります．

rebound ()

物体の座標が上下左右の壁から外に出た場合，反射するように位置と方向を設定します．

movesnake()

蛇を動かします．

168-171行

BIOSワークエリアのキーボード押下状態テーブルを直接アクセスし，左右矢印キー（カーソルキー）が押されているかどうかを調べます．左矢印キーが押されていれば角度を進行方向に対して左に1段階変更し，右矢印キーが押されていれば右に1段階変更します．

173-174行

角度からXY値への変換テーブルを利用してXY座標を更新します．

176-185行

胴体最後尾を消し，胴体の位置データを後ろへずらして新しい頭の座標を配列に格納し蛇を1歩進めます．

moveenemy()

敵を動かすルーチン．

193-202行

蛇の頭との距離を計算し，ぶつかっているかどうかを判定します．ぶつかっていれば座標データを新たに生成し，蛇の胴体を1短くします．蛇の長さが1であればゲームオーバーのフラグを立てます．

203-213行

進行方向へ座標を更新し，蛇の頭に近付くように僅かに座標を修正します．

217-218行

25分の1の確率で進行方向をランダムに変更します．

putfood ()

餌の表示ルーチン．引数の座標を中心とした蝶型の図形を表示します．

movefood ()

餌を動かすルーチン．蛇の頭とぶつかっていれば蛇の胴体を1長くし，50分の1の確率で進行方向を変化させます．

game ()

メインループ．中断キー(ESCキー)が押されていればループを抜けます．各キャラクタを動かすルーチンを順に呼び，最後にVSYNC割り込みによるカウンタ値を利用して時間待ちを行います．

main ()

パラメータが不足していれば使用法を表示し，値が不正ならばその旨を表示して終了します．初期化ルーチンを呼び出し，VSYNC割り込みをセットし，メインルーチンを実行します．VSYNC割り込みルーチンを元に戻してからプログラムを終了します．

ワンポイントテクニック

VSYNC割り込みによるゲーム速度の調整

今回は，前回の最後に予告したように，割り込みルーチンを使った時間待ちを取り上げます．前回までのプログ

ラムでは機種によるCPU速度の違いによってゲームのスピードが左右されていました．そのため起動時に時間待ち用の値を指定するなどして対応していましたが，本来ならゲームのスピードは自動的に調整したいところです．

実時間を取り入れるという意味ではシステム時計の値を得るtime関数の利用が考えられますが，１秒以下の値が得られないのでアクションゲームの時間待ちに利用するのは難しいようです．システム時計の値はもっぱら乱数初期化用の種として活躍してもらっています．

そこで一定時間ごとに発生する割り込みに目をつけることになります．PC-9801シリーズではこのような用途のためにタイマ割り込みが用意されています．タイマ割り込みは細かく正確な割り込み間隔を設定できるので便利ですが，ユーザーの使えるチャネルが１つしかないため，ほかのTSR（常駐プログラム）などがすでにタイマ割り込みを利用している場合は不都合が生じてしまいます．

一定時間周期で発生する割り込みにはもうひとつVSYNC割り込みといって，約60分の１秒ごと（CRTの垂直帰線同期信号が発生するごと）にかかる割り込みがあります．この割り込みは発生間隔を変更することこそできませんが，手軽に約60分のN秒という時間を測ることができるので，ゲームのスピード調整にはぴったりです．

今回のプログラムではカウンタを動かすVSYNC割り込みルーチンを設定し，メインルーチンでカウンタの値が一定数変化するごとに１ループ進めるといった方法でゲーム速度の調整を行っています．

では具体的にプログラムのvsync()，set_vsync()およびrestore_vsync()のルーチンを説明しましょう．

set_vsync()は割り込みルーチンvsync()をVSYNC割り込みベクタにセットします．まず安全のため割り込みコントローラのVSYNC割り込みマスクビットをセットし，元の割り込みベクタに設定されているアドレスをoldvsync()へ退避，vsync()のエントリを割り込みベクタへ設定します．VSYNC割り込みマスクをクリアし，0x64のポートへoutしてVSYNC割り込みをリセットし，最後に割り込み許可フラグをセットします．これでVSYNC割り込みが発生し，約60分の１秒ごとにvsync()ルーチンが呼ばれます．

vsync()が割り込みルーチン本体ですが，このルーチンの仕事は単にcounterの値をインクリメントするだけです．あとは元のVSYNC割り込みルーチンをコールし，0x64へout命令を出してVSYNC割り込みをリセットします（毎回このポートを叩いてやらないと次のVSYNC割り込みが発生しない）．Cプログラムから見ると通常の変数であるcounterの値が定期的に変化することになり，コンパイラのオプティマイズによっては264行目のプログラムが意図した動作をしない（counterの値が変化しないものとして永久ループになってしまう）ことがあり得るので，注意が必要です．

最後にrestore_vsync()で割り込みベクタを元に戻します．ここでは割り込みコントローラをマスクしたままになっていますが，ほかのVSYNC割り込みを使用するTSRを使っている場合には最初のマスク状態を保存しておいて同じ状態に戻すようにしたほうがよいでしょう．

このように，CプログラムからのVSYNC割り込みの利用は比較的簡単な手順で実現できるのですが，いくつか注意しなければいけない点があります．

まず第一に，いったん割り込みベクタを自分のプログラムのエントリと入れ換えたら，プログラムを終了（中断）する前に必ずベクタを元に戻すルーチンを呼ばなければいけません．今回のプログラムではstopキーが押されたときの対策をしていないので，実行中にstopキーを押すとプログラムが中断してしまい，再度起動すると暴走してしまいます．

もしVSYNC割り込みのルーチンでもっと時間のかかる処理をしたかったら，さらに手順を増やさなければいけません．割り込みルーチン内では使えるスタックの大きさが保証されていないので，一時的にSS，SPを変更してローカルスタックを設定する必要があるし，処理に時間がかかるようならほかの割り込みの発生を許可しなければいけません．万一，割り込みルーチンが終了する前に次のVSYNC割り込みが発生しても暴走しないように，再入時の処理（割り込み処理中はフラグを立てておき，フラグが立っているときに同じ割り込みが発生したらすぐに戻るなど）にも対応しておきます．

割り込みルーチンはとても便利ですが，通常のプログラムとは違った条件で動作をしているため，それなりの注意が必要なことがわかってもらえたことと思います．割り込みルーチンについては，常駐プログラムを取り上げるときにもう少し詳しく説明したいと思います．

パソコン博物館 XI

新技術の開拓者 FM8

富士通 FM8

井 戸 禎 光

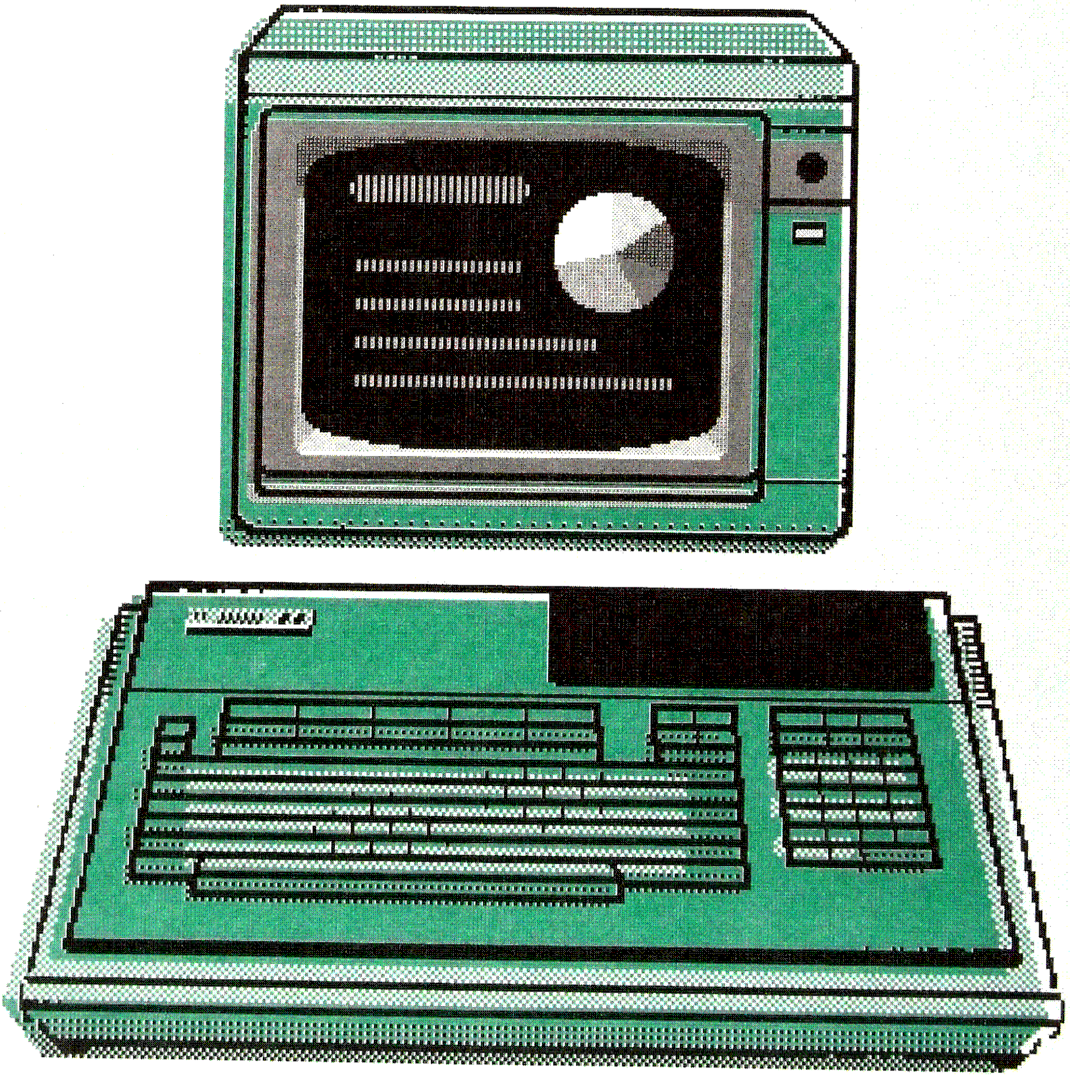


Illustration
TOSHIHITO ITO

富士通がパソコン市場へ参入したのは割合遅く，1981年5月のFM8（FUJITSU Micro8）でのことだった．CPUは，8ビットの68A09×2で，RAMは64KB，価格は218,000円であった．

このマシンは，重装備との評判であった．メインCPUはZ-80に切り替えることも可能で，OSもCP/M，OS-9，FLEX09から選べるようになっていた．もっともCP/Mで走らせるためには，Z-80カード（11,700円）を中に入れる必要があった．補助記憶装置で変わったものとしては，磁気バブルメモリというのがあった．メモリでも64KビットDRAM（MB8264×8）の採用を大々的に詠い，64KBの容量をもっていた．

この時期に競合していたマシンとしては，MZ-80B，if800M30，ベーシックマスターレベル3，PC-8801などがあった．一方，この年，米国IBMおよび三菱電機は16ビット機でパソコン業界に参入している．

そのような環境において，汎用コンピュータのトップ企業，富士通の対応に興味が集まっていた．その答えがFM8であり，同年に発表されたFACOM 9450（120万円）であった．9450のほうは，主としてビジネス用であった．作りもワンボードタイプではなく，メモリ，CPU，I/Oなどのいろいろな機能に分かれて作られている6～7枚のボードをたくさんのコネクタのついたバスボードに差し込むような形式に作られていた．

さて，FM8であるが，こちらのほうは，218,000円という，当時のパソコンの標準的な価格設定の中で，どこまで新しい半導体部品を使って高機能を盛り込めるかという，技術展開のスタープレーヤーであった．その例をあげれば，CPUチップの採用方針である．当時すでに，インテル系のCPUであるZ-80が非常にポピュラーになっており，市場で入手可能なソフトウェアの多くはZ-80対応であった．しかし，CPU速度の点においては，モトローラ系の6809が後から出たこともあって，とくに高速で究極の8ビットCPUといわれていた．富士通は高速機能を重視して選定したといえよう．

FM8は，68A09を2個使っていた．そのうちの1個は，当然CPUであるが，もう一方はキーボードとディスプレイ専用に使っている．キーボードは本体に組み込まれているが，その制御には別に4ビットのマイクロコントローラMB8841を用いている．ROMとしては漢字キャラクタ・ジェネレータ（オプション）用に64KビットのMB8364を16個，F-BASIC用にも同じくMB8364を4個積んでいる．メインメモリにおいても，当時最新の64KビットDRAMを使っていたし，またそのことを大いにPRしていた．

漢字ROMについては，当時何に使っていたのかというと，1文字ずつ必要な漢字を拾っていって，何とか漢字で表示ができるということで，大いに満足していたものである．それ以前のパソコンは“BASIC”を表示していることが多く，ほとんどCRT全面に英文字・数字が表示され，ところどころにカタカナが入っているのみという状態で，非常に見にくかった．ひらがな・漢字がCRT上に出せるなどということは夢であった（英数字とカタカナだけでプリントアウトされたドキュメントを想像してみてほしい）．したがって，未だ漢字ROMはオプションでよかったのである．その後，このようなハードウェア環境が整った後に，日本語ワープロの全盛時代が約束されていた．

このような点は，初期のどのメーカーにおいても共通することであるが，パソコン事業は半導体部品の需要開拓の一環としての任務を帯びていたことの現れでもある．扱いは半導体統括営業部となっていた．この点は，必ずしもコンピュータとしての総合性能にまずいことがあったわけではなく，パソコンの性能がパワフルになっていくよいきっかけになっている．

さて，現在の富士通であるが，1981年にパソコン業界に参入した時と同じくFMRシリーズとFM TOWNSシリーズの2系統で健闘している．ただ，当時と異なるのは，主としてビジネス用に向けられているFMRシリーズは，ホビー用のFM TOWNSよりずっと価格が高いというわけではないことである．

FM TOWNSは，富士通がコンピュータの第一人者としての面目を保ち，新しい技術の流れとしての“オーディオ・ビジュアル”技術を取り入れたパソコンとして，わが国初の製品といってよいであろう．CD-ROMは，世の中にあふれているわけではないので，まだ爆発的な売れ行きとはいかないかもしれないが，“新分野は誰かによって開かれなければならない”のである．

祐安重夫

プログラマの道具学

第11回 コーヒー

Illustration
Mikio Sasaki

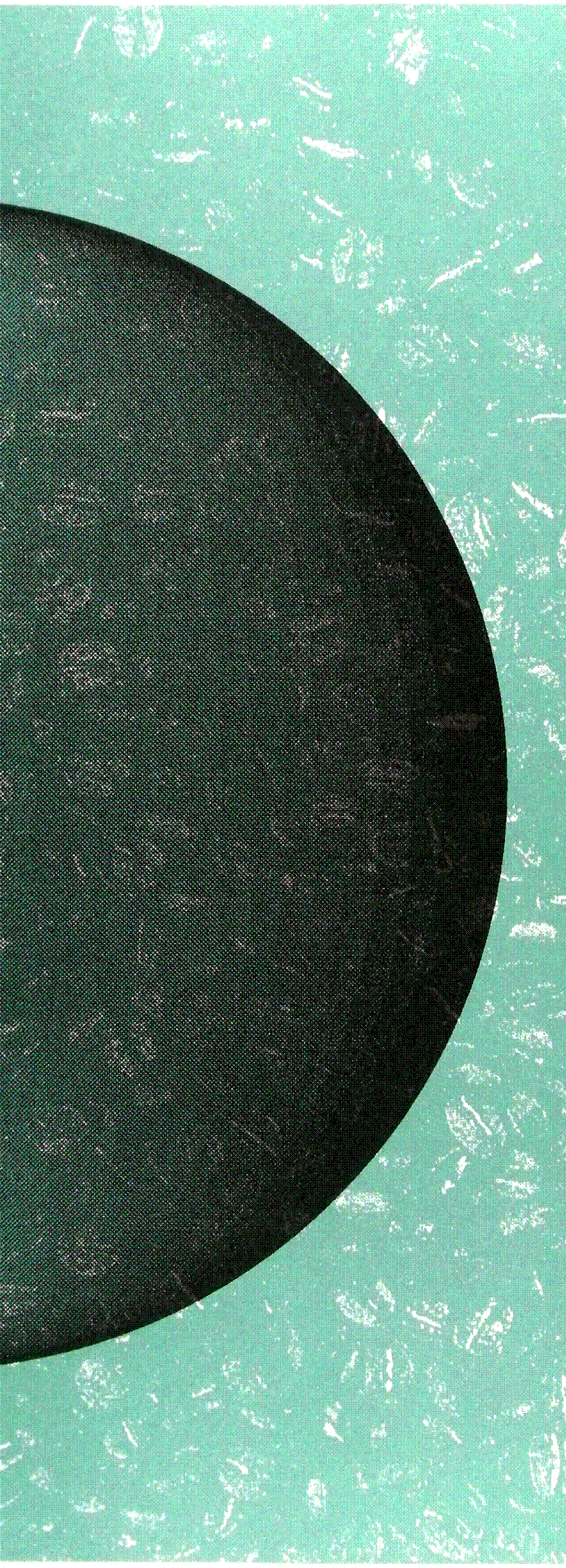
コーヒーというものを意識的に味わってのむようになったのは，今から20年以上も前の高校3年の頃だっただろうか．たしかコーヒー専門のチェーン店がすこしづつ出現してきたあたりで，サイフォンでいれたストレート・コーヒーを，砂糖もクリームもいれずにブラックでのむ習慣がついた．すぐに自分でもサイフォンを手にいれてきて，家でもわかしてのむようになった．

それからすこしして，ネルの布でいれたドリップ・メイキング・コーヒーにであって，そのうまさにおどろいたが，さすがにこれは自分でこころみようとはおもわなかった．道具は単純なものだが，ほっておくと雑巾のようなにおいがしてきて，その状態を管理するのが大変だったからである．サイフォンについても同様の問題点があって，結構苦労した．

しばらくするうちにペーパー・フィルタが出現して，コーヒーの抽出は誰にでもできる簡単なものになった．さらにこの技術をつかったコーヒー・メーカによって，家庭やオフィスでいつでもいれたてのコーヒーがのめるようになったのである．私が最初にコーヒーを味わうようになるきっかけとなったコーヒー専門のチェーン店は，最初はサイフォンをつかっていたが，現在ではペーパー・フィルタでいれるようにかわってしまった．

ペーパー・フィルタでいれたコーヒーは，実際にはそれほどうまくない．ためしにコーヒー豆をいれずにお湯をフィルタにとおしただけのものをつくって，のんでみればわかることだが，たぶんフィルタにふくまれる糊質がとけだしてくるようで，異様な味のお湯になる場合がある．

もっとも最近のペーパー・フィルタは，紙の貼付に糊を使用せず圧力をかけることでおこなっているものがふえている．また一部だが，漂白していない薄茶色のペーパー・フィ



ルタもあって，自然食品店などで手にいれることができるので，最近ではもっぱらそれを購入している．

今ほしいとおもっているのはエスプレッソ・マシンとウォーター・ドリップ用の器具なのだが，本格的なエスプレッソを抽出するには，喫茶店用の高価な機械を必要とするようなので，将来この仕事を引退して，喫茶店でもひらくときのたのしみにとっておこう．プログラマの老後の対策については，今のところまだノウハウが蓄積されていないので，自分でかんがえておいた方がよさそうである．

ウォーター・ドリップについては，何度か道具をつかわずにつくってみたことがあるが，濾過に大変な手間がかかるので，今ではほとんどやっていない．最近になって家庭でウォーター・ドリップができるような，手軽なガラス器具がいくつかでてきたので，今年の夏はひとつ購入してみようとおもっていたのだが，もう夏もおわりになってしまった．

仕事場にはコーヒー・メーカがおいてあって，普段はそれでいれたコーヒーをのんでいるが，コンピュータにむかいながらのむコーヒーは，気分転換か眠気ざましといった目的が主になるので，あまり味にこだわらずに妥協している．本当にうまいコーヒーがのみたいときにでかける店が何軒かあって，これはあまり人におしえないで，自分だけのたのしみにとってある．

某国立大学ではイーサネットにコーヒーサーバをつないで，プリンタサーバやファイルサーバと同様にワークステーションからコントロールしようという計画がすすんでいるらしいが，そこまでの要求はうちの仕事場ではまだない．

そういえばときどきキーボードにコーヒーをこぼしてしまったという話をきくことがあるが，さいわいまだそれは経験していない．

IC千夜一夜

第5回 メモリのIC化

D-RAM LSIの発展

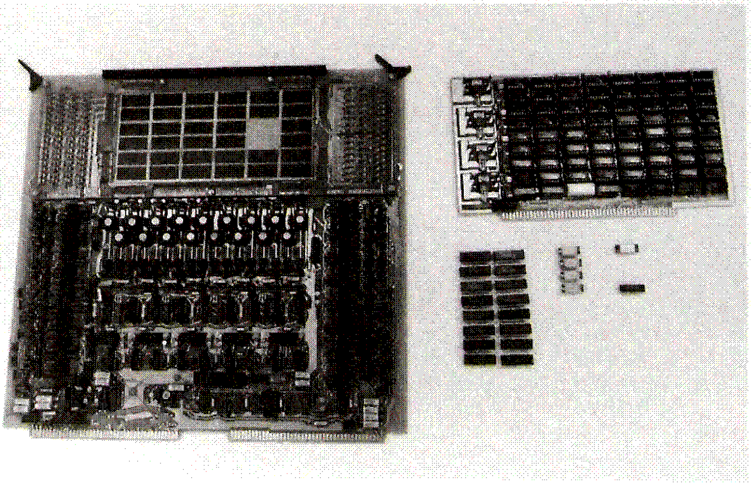
虎岩 登

前回に引き続き，メモリについて眺めていきます．今回は，IC化されたD-RAMを取り上げます．

メモリをIC化すると，どこがよくなるのでしょうか．実際のところ，悪くなる点はほとんどないというべきでしょう．コアメモリは，磁気記憶の一種なので，電源を切っても内容が残る－不揮発性と呼ばれる特徴がありました．この点以外は，すべてよくなるばかりだといえるでしょう．大きさ，スピード，コスト，容量，どれをとっても大幅に改善されてきています．最後まで残っていた不揮発性という点でも，最近では半導体メモリICの発展によって，十分な信頼性を持ったものが，安価に供給できるようになっています．

どれだけよくなったのかを，大きさの点で実感していただきましょう．写真は，約8KBのメモリが，どれだけの大きさになるかを示したものです．コアメモリ（1970年代の製品で，コアメモリとしては高い集積度のものだと考えられます）では，50cm四方もの大きさだったものが，最近では1個のICの64分の1，マッチ棒の頭ほどの体積も占めません．約70，000個ものコアがコアメモリの基板上にあるのだというと，かえってコアメモリの意外な小ささに驚かれるかもしれません．多分，肉眼ではコアの1個1個は判別できないと思います．こんなに小さなコアメモリでも，IC化されたメモリにははるかに及ばないのです．

写真1　コアメモリ（左），S-RAM 1Kbit（右上），D-RAM（右下左から4Kbit，16Kbit，64Kbit，1Mbit）



ダイナミックメモリ
データ記憶方式と内部構成

ダイナミックメモリ(以下D-RAM LSI)は，コンデンサに蓄えた電荷で，メモリ動作を行います．コンデンサというのは，日本語でいえば蓄電器です．電気を蓄えるという機能では，ノートブック型コンピュータに使われているNiCd電池のような蓄電池とよく似ていますが，両者は動作原理が異なっていますので，充・放電のスピードや寿命が大幅に違います．コンデンサのほうが，はるかに高速で，寿命も永久的です．ノートブック型コンピュータの電池は，交換する必要に迫られている方がいるかもしれませんが，メモリはまず壊れません．ただし，貯められる電気の量という点では，メモリICに入っているコンデンサは大きさが非常に小さいので，通常の電池とは比較にならないほどわずかな量しか入りません．この小さなコンデンサが充電されているかどうかで，記憶を行います．

コンデンサに電気を溜めても，少しずつ勝手に放電してしまいますから，そんなに長い時間記憶しておくことはできません．ですから，先月の遅延線メモリと同じように，一定時間内に再書き込み(リフレッシュ動作)を行うことで，永続的な記憶素子として利用します．これが「ダイナミック」という名前の由来なのです．

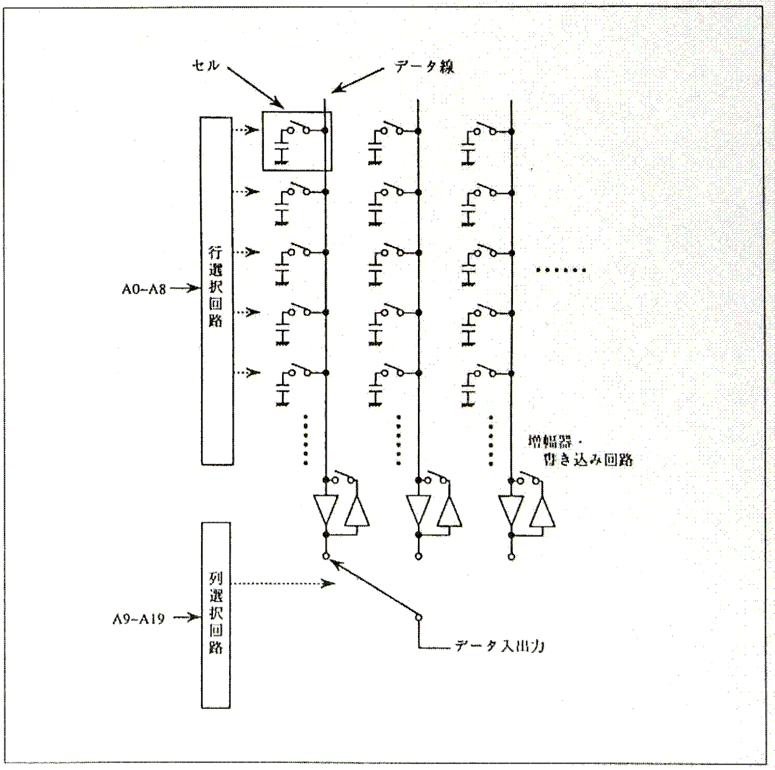
規格では，2ms～8msの時間内にリフレッシュを行うことになっています．リフレッシュしなくても，常温付近では数秒くらいは楽に記憶を保持できます．リフレッシュの時間は短すぎるようですが，温度が高くなるにつれて急激に放電しやすくなりますし，記憶内容が消えてしまうことはコンピュータにとって致命的な欠陥ですから，十分な安全性を見込んでいるのです．

現在のD-RAM LSIの内部の概略は，図1のようになっています．コンデンサ1つとスイッチ1つの組み合わせで，1ビット分の記憶が行われます．この記憶単位をセルと呼びます．先月号で，ランダムアクセスを行うときは，スイッチを2段階に組み合わせて，スイッチの総数を減らすテクニックが用いられることがあると説明しましたが，ここでも似たようなことが行われています．

たとえば1MビットD-RAM LSIの場合ですと，セルが1本の電線(データ線)に512個並列に接続されています．この電線が，2048本チップ上に並べられているのです．アドレス信号のうち，上位9ビットで，セルの内部の512個のスイッチのうちの1つを選択して閉じます．そうすると，各データ線につき，1個のセルの内容だけが出力されます．このとき，コンデンサが非常に小さいので，信号も小さなものになってしまいますが，データ線ごとにそれを増幅して十分な大きさにします．それから，アドレスの下位11ビットで1本のデータ線を選択します．この場合，各セルのスイッチ数は減っていませんが，スイッチに対する司令を送る電線の本数が劇的に減っていることになります．

このような基本構成を利用して，さらに便利に使えるよう，さまざまな機能が付け加えられています．まずアドレス信号は，上位9ビットと下位の11ビットは同時には必要ありません．第1段階の512個のセルから1つを選択する操作をしてから，次にデータ線のうちの1本を選択するまでには，信号がデータ線を伝わって増幅されるまでの時間に遅れがでます．ですから，データ線を選択する下位11ビット分は，上位のアドレス線より少し後に入力し

図1　D-RAM LSIの内部構造(タイミング発生部など省略)



ても間に合います．そこで，パッケージを小さくして高密度実装を可能にするために，アドレス信号は上位と下位の２回に分けて入力するようにしています（実際には下位アドレスのうちの１ビットだけは上位アドレスと同時に入力します）．このようにすれば1MビットD-RAM LSIでは，本来20本のアドレス信号が必要なところを，アドレス端子は10本で済ますことができて，28ピンのパッケージを使う代わりに18ピンの小さなパッケージが使えるのです．

データアクセス方式
高速アクセスの工夫

D-RAM LSIの内部では，１度に2048ビット分のデータが読み出されています．このうちの１ビットだけを読み出して，残りは捨ててしまうのはもったいないような気がしますね．そこで，近接したアドレスの内容を高速アクセスするモードがつけられています．同時に読み出されているデータなら，いちいちセルの小さな出力を増幅する必要もなく，出力部分の切り替えスイッチを操作するだけで出力できるため，高速アクセスが可能なのです．最近の高性能マイクロプロセッサは，バーストサイクルなどと呼ばれる連続した数ワードの命令を一気に読み出す機能がありますが，これはD-RAM LSIの近接したアドレスを高速で読み出すこの機能を見込んで設計されているわけです．近接アドレスの指定方法によって，ページモードとかニブルモード，スタティックカラムモードなど，いくつか種類がありますが，原理的には同じようなものです．グラフィックディスプレイなどに応用されるデュアルポートメモリも，同時に多数のビットが読み書きされる性質を有効利用している特殊なD-RAM LSIです．

アクセス方法の工夫としては，ほかにリード・モデファイ・ライト・サイクルというものもあります．D-RAM LSIは，一般的に破壊読み出しです．読み出すときには，データが記録されているコンデンサの内容を放電させてしまい，記憶がセル内部からは失われてしまいます．したがって，読み出した後に，必ず自動的に書き込みが行われます（そんなわけで，リフレッシュ動作も一種の読み出しを行わせているのです）．そうすると，単に用意したデータを書き込む方法のほかに，いったんデータを読み出してから再書き込みを行うときに，別のデータに入れ替えてしまう方法が考えられます．

1MビットD-RAM LSIの場合だと，セルへの読み書きは必ず2048本のデータ線ごとに同時に行われますから，１ビットを書き込もうとしても，残り2047本の書き込みデータを用意するために，内部的には１度読み出しを行わなければなりません．つまり，同じアドレスに対する読み書きを続けて行うなら，外部に読み出さずに書き込むのと大して変わらない手間しかかかりません．これが時間の節約になるのです．

この機能を上手に使えば，メモリに対するインクリメント命令などを高速にできます．メモリの遅かった時代のミニコンなどにはよく使われていましたが，現在では，メモリも十分に高速になったためか，あまり使われません．

また，D-RAMの欠点でもありますが，読み出しの後に再書き込みをする必要と，次の読み出しに備えてデータ線の電圧を一定の電圧に戻すために，内部ではある一定の時間が必要です．なにしろ，コンデンサから出力される信号は非常に小さいので，読み出す前にデータ線をHiかLowかの境目の電圧にしておかないと，増幅して判定することができないのです．アクセス時間の割に，連続アクセスのときには内部での準備時間が必要なので，それほど短時間にはできません．

メモリにアドレスを与えて，データが出てくる時間をアクセス時間，１回読み出しを行ってから，次に読み出しが可能になるまでの時間をサイクル時間と呼びます．D-RAM LSIでは，「サイクル時間＞アクセス時間」なのです．S-RAM LSIでは，このような後始末や準備の時間は必要ありませんから，「サイクル時間＝アクセス時間」になります．このような欠点を補うためにも，先に説明したようなさまざまなアクセス方法が用意されているわけです．

メモリICの歴史
256ビットから4Mビットまで

IC化されたメモリを，きわめて熱心に開発・販売していたのはインテル社です．いまでこそ，86系のマイクロプロセッサの供給元として知られていますが，マイクロプロセッサを開発する前から，メモリICを生産していたのです．もちろん，現在のメモリに関する技術はほかの企業も多く寄与しています．

初期にベストセラーになったものには，256ビットのスタティックメモリ1101や，1Kビットのダイナミックメモリ1103がありました．ともに，p-MOSプロセスを使用しています．したがって，電源が5V単一ではなく，消費電力の大きい割にはあまり高速とはいえませんでした．試みに，1101で8KBのメモリボードを製作したとすれば，写真１のコアメモリボードと大きさとスピードでは互角，価格や消費電力ではコ

アメモリのほうが優れていたはずです．もちろん，コアメモリを確実に動作させるには，アナログ回路的なノウハウと複雑なタイミング設計が必要で，その分だけ楽になっていることは確かですが．

D-RAM LSIが同時代のS-RAM LSIと比較して大容量だというのは，1ビットのデータを記憶するセルの構造によります．S-RAM LSIは1セル当たり6素子が必要ですが，現在のD-RAM LSIではコンデンサと，スイッチ役のトランジスタの，2素子しか必要ありません．もっとも，D-RAM LSIが生まれたころはもっと多く，1Kビットの1103では4素子のセルが使われていました(図2)．現在の構成では，データ線に対し1つのアンプが付いていますが，初期のものではセルの1つ1つにアンプを入れていたのです．

また，4KビットのD-RAM LSIの時代には，Mostek社がMK4096というメモリを開発しました．ある意味で，現在のD-RAM LSIの形を作ったICで，アドレスを2回に時分割して入力する方式になったのはこれからです．それ以前の4Kビットの2107では，アドレスを時分割していないので，22ピンのパッケージですが，MK4096は16ピンです．また，2素子型のセルやすべての入出力がTTLレベルであるなど，著しい改善が行われています．2107には，0Vから12Vまで高速に変化させなければならない入力がありました．

16Kビットの製品は，技術的にはMK4096の素直な延長線上にありました．ただ，この16Kビットの旬の時代は，ちょうど初期のパーソナルコンピュータが立ち上がる時期だったので(PC-8001, APPLE IIやPET, TRSなどです)，これを採用する製品が多く，D-RAM LSIが身近になってきたのを感じられたものです．

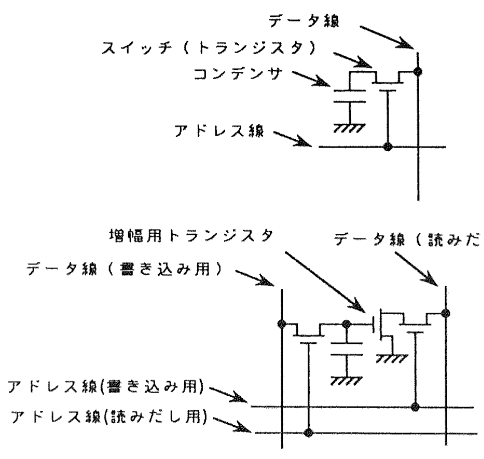
16Kビットの製品までは，3種類程度の電源が必要でした．通常のCPUなどが5Vの電源だけですむようになっても，D-RAM LSIの場合は内部回路の都合のためか，最後まで複数の電源が使われていました．しかし，64Kビットになると，5V単一電源を使うようになりました．どうしても必要な負電源などは，チップ内部で作り出す技術を使ったのです．また，この時代からリフレッシュを容易にする工夫を入れたり，ニブルモードなどの高速アクセス法の導入が行われました．

さらに，1つのチップで4×16Kビット構成，つまり1つのアドレスに対し4ビットのデータを記憶できるものなどバリエーションが増えました．これ以前は，1×64Kビットのような構成のものしかありませんでしたから，バイト単位のメモリを作るためには，64Kビットチップを8個集めて，64KBのメモリにするのが最少のメモリでした．4×16Kビット構成なら，2個で16KBのメモリができます．だんだん，マイクロコンピュータ用メモリとしての応用が重要視されてきたのですね．

256Kビット，1Mビットと世代が移って，現在は4Mビットが旬になりかけているといったところでしょうか．1Mビットのものでは，回路構成がC-MOS化されて消費電力が減りました．また世代ごとにタイミングにやスピードに改良が加えられ，SIMMというモジュール構造の製品など，実装方法にも工夫が見られます．

昔は，「256Kビットはいつ頃までに開発される」とか，「1Mビットは何年頃か」という予想を見て，そんなに大容量のメモリを誰が何に使うんだろうと想像したものですが，4Mビットの時代ともなると，Windows 3.0とかSystem 7とか，いくらでもメモリをつかうソフトウェアができているものなのですね．

図2　セルの構造



現在使われている1トランジスタ・1コンデンサ型セル(上)と、1Kビットの1103に使われた3トランジスタ・1コンデンサ型セル（下）．後者は、配線の本数も2倍必要だった．

MS-DOS TOOL LABO

MS-DOSツール・ラボ

第8回 リングバッファの利用・起動記憶プログラム TIMERING.EXE

加野島英渡

今回は，キーボードバッファ管理や通信のための送受信処理などによく使われるデータ構造である，リングバッファについて解説を行う．実際のプログラム例としては，10回まで起動日時を記憶するツールを作成した．

秋の気配と夏の名残り

ここ札幌は早くも秋まっさかりである（まぁ，いくら北海道だからといって，今の時期にスキーができるなどということはないが）．気候も過ごしやすいし，食べ物も旨いし，短い北国の秋のひとときを満喫している．暑さの厳しい地域では，未だ夏のダメージが残っている方々もいらっしゃるかと思うが，ご自愛いただきたい．さて，先月まで勢いにまかせてコンパイラの作成まで走ってしまった．あまりペースを上げすぎると，夏の疲れが残っている体には少々重くなってきたかということで，このへんで一息入れて今月はちょっと別の話題をお送りしたいと思う．

遊園地にて思うこと

行楽のシーズンである．子供にせがまれて（あるいは自分が行きたくて）遊園地へ足を運ぶ人も多いかと思う．数ある遊園地の遊具の中でも，ひときわ目をひくものに「観覧車」がある．そう，あの巨大な輪にゴンドラがいくつもぶらさがっていて，のんびりと回っている「観覧車」だ．近頃は，よくもまぁ人を怖がらせるネタがこうも出てくるものだと思うくらい，次々と登場するジェットコースターの類にその主役の座を奪われているとはいえ，その堂々とした威容は遊園地の華と呼ぶにふさわしい（と，怖いものが好きではない筆者は勝手に考えている）．

さて，その観覧車の「乗り口」のところに長い行列ができているとしよう．「降り口」のほうでは，1周で数分の空中遊覧を楽しんだ人達が降りてくる．「乗り口」の行列から空になったゴ

ワンラインNEWS
ソフトビジョンはブラインドタッチ練習ソフト「TYPE RUNNER」を発売した．指定した指ごとに練習ができ，不得意な指をグラフで確認することができる．

ンドラの分だけ，新しく人が乗り込んでいく．料金をたくさん払うと，何周もできるシステムがあるのかどうかはわからないが，普通なら降りてくる人はその時点でいちばん先に観覧車に乗った人である．これはまさに「First In First Out (FIFO)」のデータ構造になっているではないか(遊園地でこんなことを考えているのは，やはりあまり健全ではないが)．

環状データ構造のこと

この観覧車が，ああも優雅に回り続けていられるのは，その環状になった構造のおかげである．もし，これが直線状にゴンドラが取り付けられていれば，最初に乗った客を最初に降ろすという処理をスムーズに行うことはできないだろう．たとえば，「お猿の電車」(こういうものがまだあるのだろうか？）ならば，一旦停車してすべての乗客を入れ替えてまた出発する．

ここで話はいきなりコンピュータに飛ぶが，プログラムで「いちばん先に入れたものをいちばん先に出す」つまりFIFOのデータ処理をすることは少なくないが，その場合も観覧車のように環状になった構造をとると，処理がしやすくなることがある．

この種の処理の典型的なものに「キーボードバッファ」がある．たいがいのキーボード入力処理は，

- ハード的な割り込みによって，押されたキーのコードをバッファに格納する．
- プログラムからの要求によって，キーバッファの内容を取り出す．

この2つからなっている．押されたキーは，ハード的な割り込み処理でキーバッファに格納され，取り出しはプログラムからの要求によって処理されるため，この2つの処理は非同期で行われることになる．円滑にFIFOの処理が行われないと悲惨なことになるのは容易に想像できる．

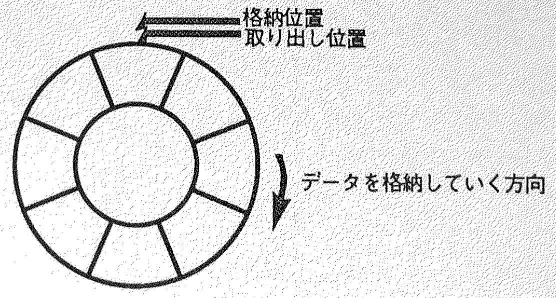
そこで，この機能を実現するためのデータ構造を考えてみよう．

キーボードから「D」「I」「R」と3文字を入力して，その最初の1文字「D」を取り出す処理を考えてみる．もしデータバッファが，通常のように開始アドレスと終了アドレスが決まった直線構造をしていると，図1のように，どうしてもデータの格納・取り出しの際にデータの移動が必要になってくる．ポインタを用いて，データを移動させずに格納・取り出しすることもできなくはないが，それをするためには，バッファの空き領域の管理がとても複雑になってしまう．

この点を解消するために，同じ処理をするためのバッファを環状に取ってみる．いま一度観覧車を思い出していただきたい．乗り口から乗った客（この場合はデータ）は，ゆっくりと上のほうへ(結果的には降り口に向かって)移動していく．プログラムの場合，バッファ自体は回さないで固定しておき，「乗り口」「降り口」が移動するのだと考えればいい．データが1つ乗ったら乗り口（「格納位置」と呼ぶことにしよう）は1つ移動して次のデータ格納位置へいく．降り口（こちらは「取り出し位置」と呼ぶことにする）も，1つデータを取り出すごとに次に取り出

図1

図2



すべきデータ位置へと移動していく.

図2の例だと，初期状態は格納位置・取り出し位置が同じところにある．データはまだ何もない．

この状態から，最初の一文字を格納すると次のデータのために格納位置は1つずれる．取り出しはまだなので取り出し位置は変化しない（図3）．

この要領で次々とデータを格納していくと，格納位置がそれに合わせてずれていく．このときに，もし格納位置が取り出し位置に追い付くことがあれば，データがバッファ1周分すべて格納されたことになり，領域はいっぱいということである(図4)．

今度は，データを取り出す場合を考えてみよう．取り出しは，取り出し位置が示している場所から必要な分だけ行えばよい．その後で，次の取り出しに備えて取り出し位置を先へ進めておく．取り出し位置が格納位置と同じ場所を指していれば，データが1つもないということなので当然取り出しは行えない(図5)．

このような手順で，データを格納して取り出していけば，データの位置を示す指標を移動するだけでデータの移動は必要なくなる．格納するときは格納位置をずらし，取り出すときは取り出し位置をずらすという処理だけで，2つの指標は永遠に1つの円周上を回っているわけである．ふたつの指標の動きをまとめると以下のようになる．

・格納位置が示す場所に格納する．その後格納位置を更新する．
・取り出し位置が示す場所から取り出す．その後取り出し位置を更新する．

《ただし》

図3

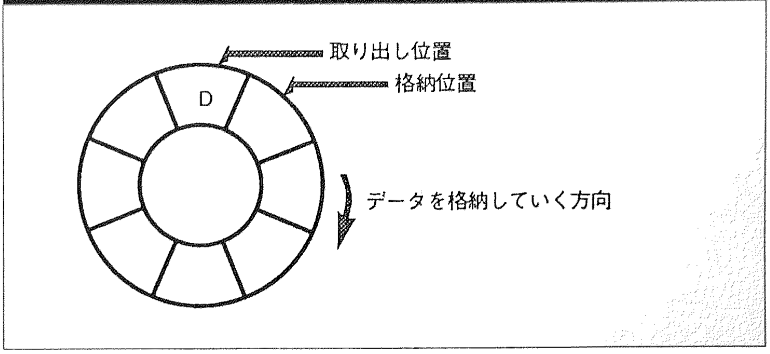


図4

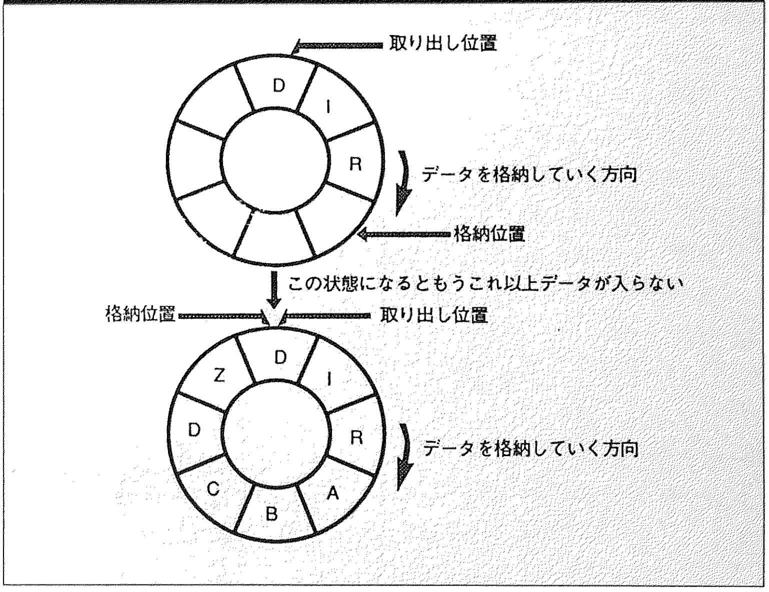
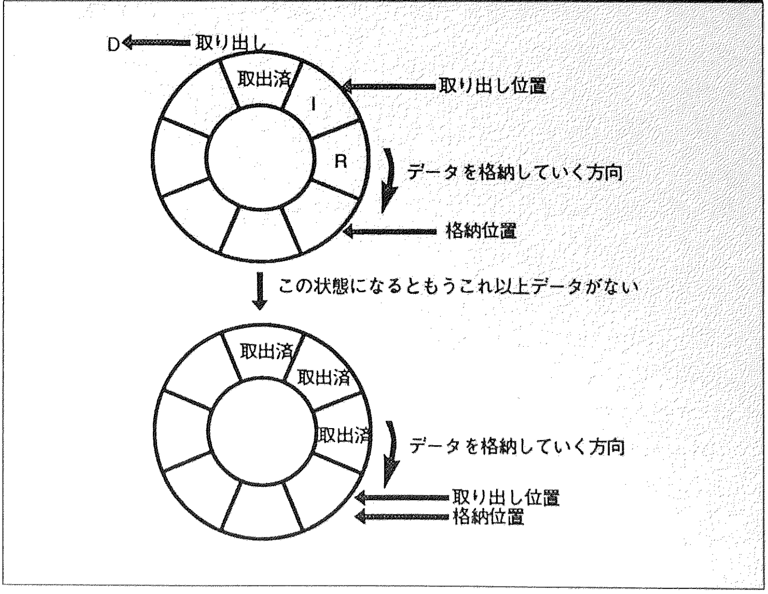


図5



ワンラインNEWS
日本IBMが'89年11月から建設工事を進めていた「日本IBM幕張ビル」が完成した．同ビルは顧客のコンピュータシステム構築の支援と，教育拠点としての役割を担う．

・取り出し位置が格納位置に追い付いたらデータが空になっている．
・格納位置が取り出し位置に追い付いたらデータが満杯になっている．

ところで，この「環状」というのはあくまでも概念上の話であり，コンピュータのアドレスをこのようにすることはできない．ではどうするかというと，「取り出し位置」と「格納位置」を更新するときにちょっとした操作をするわけである．

「位置を更新しようとしたときにバッファ領域の最後までいってしまったら，バッファ領域の先頭をさすようにする」．

具体的にいうと，データ格納領域が10バイトあって，その領域の先頭アドレスが2000hだった場合，位置の更新はリスト1のようなコーディングになるだろう．

そしてリングバッファ

環状になったデータ構造の利点がおわかりいただけただろうか．当然ながら，別にこれは筆者が遊園地で考えついたものなどではない．この構造は，環状になったデータバッファ構造ということで，一般に「リングバッファ」と呼ばれており，プログラム手法ではお馴染みのものである．前記のキーボードバッファ管理がその典型であるし，通信のための送受信処理などでもよく使われる手法なのである．

では次に，実際のプログラムでどのようにこのリングバッファを管理するかを3つのタイプに分けてまとめてみよう．

■①リングバッファの典型的な例

リングバッファを管理するための代表的な手法では，「取り出し位置」と「格納位置」という2つの指標を用いる．ここでは格納位置のことを「ヘッドポインタ」，取り出し位置のことを「テールポインタ」と呼ぶことにする．格納データの先頭(最後に入れたデータ)を示すのがヘッドポインタであり，最後尾(最初に入れたデータ)を示すのがテールポインタであるという考え方である．そして実際のプログラムの中では，ヘッドポインタは次にデータを入れる位置を示し，テールポインタは次にデータを取り出す位置を示す．

では，具体的にこの2つのポインタの動きを見てみよう(図6)．

■②データカウンタを併用する管理手法

①に書いたポインタを2つ使う手法が，リングバッファの代表的な管理方法であるが，これには少々問題点がある．というのは，2つのポインタがり

リスト1

```
    ;------ 位置の更新
        INC     [位置]                  ; 位置を更新してみて
        CMP     [位置],2000h + 10       ; バッファの末端まで達したか？
        JC      INC_OK                  ; そうでなければそのまま
        MOV     [位置],2000h            ; 末端に達した場合は先頭へ
INC_OK:
            .
            .
            .
```

COLUMN　名探偵気分でデバッグを……

プログラムのデバッグをしていると，ときとして推理小説を読んでいるような気になってくることがある．まず，全体の動きを見て(これが犯行現場である)，ルーチン(容疑者)を絞っていく．1つ1つのルーチンの動作を確認したり(アリバイ探し)，そのルーチンの上位・下位モジュールとのインターフェースを検証したり(身辺の聞き込み)……，そしてアタリをつけたロジックにとうとうバグを見つける(逮捕)．

ミステリー仕立てに考えれば，バグを見つけるまでに伏線も用意されているだろうし，いかにも怪しそうな雰囲気をもった「おとり」もうようよしている．複雑に絡まった糸を少しずつ解きほぐしていって，核心に迫っていく過程にスリルを感じられるようになれば，もう気分は名探偵である．

これが多人数での開発となると，文字どおり「犯人探し」になることもあるが(人間関係に亀裂が入りやすい瞬間である)，自分で作ったプログラムを1人でデバッグしていても，この雰囲気は十分に味わうことができる．

少し経験を積んでくると，捜査(？)にも「第六感」が働くようになってくるのだ．

「あ，これはスタックがずれている」
「あ，このへんでレジスタを壊している」
「あ，ADDとSUBを間違えた」

地道な検証ももちろん大切だが，こういう勘によって劇的にデバッグが進行することもあるのが，プログラミングの醍醐味ではないかという気さえしてくる．デバッグに行き詰まったら，パイプでもくゆらして古今の名探偵を気取ってみるのも，気分転換にいいかもしれない．まぁ，現実には「安楽椅子探偵」というわけにはいかないだろうが……．

ング状に変化していくために，ある時点ではヘッドポインタがテールポインタより低位（小さいほう）のアドレスに位置しているということが発生する．このため，格納するときに，あと領域がどのくらい残っているか，また取り出すときにはデータがどのくらいあるかの判定が非常に複雑なものになってしまう．

こうした場合，格納するデータが固定長であれば，2つのポインタに加えてデータカウンタを併用したほうがより効率のいい管理ができるだろう．バッファ領域を決めたときにおのずと格納できる最大個数も決まるので，格納する際には最大個数のチェックさえ行えばテールポインタがいまどこにあるかを気にする必要がなく，取り出すときもカウンタの数だけ取り出せばヘッドポインタを参照する必要がなくなるわけである．

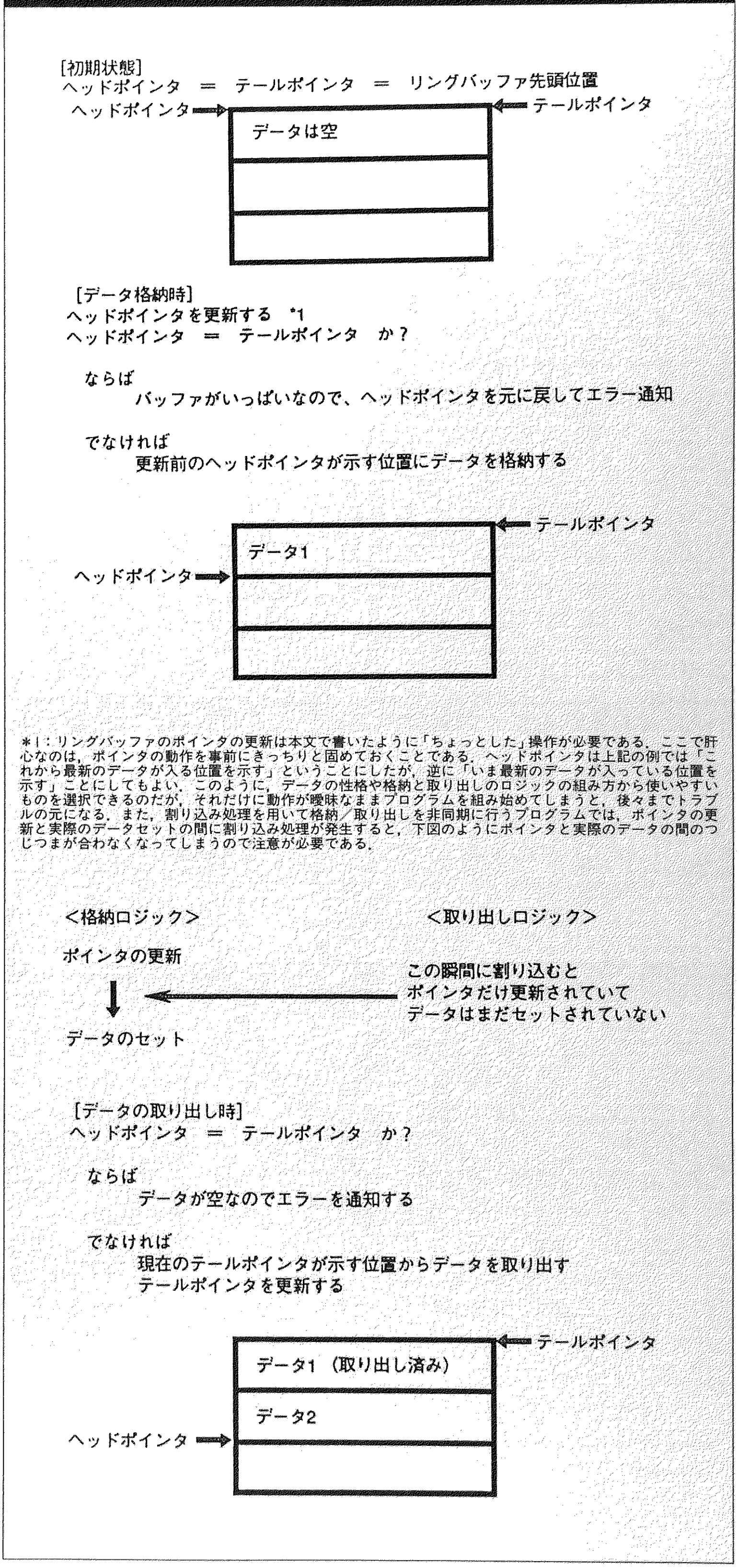
図7を見ると，先に説明したヘッドポインタとテールポインタのみの管理よりロジックがかなりシンプルになっていることがわかる．とくに図8のように，ヘッドポインタとテールポインタの位置が逆転している場合にこの方法のメリットが効果的になる．

■③単純化したリングバッファ

3つ目は，もっとシンプルにリングバッファを使う手法である(図9)．ポインタは1個しか必要ない．ただし使いみちは限られてくる．

- 最新の何個かのデータを保存する．
- 最大個数に達している場合は，いちばん古いものの上に上書きしていく．
- バッファ上に記録されている全データを「古い順」または「新しい

図6



ワンラインNEWS
ウェスコムは通産省の外郭団体である情報処理振興事業協会の委託で「パソコンの画面を見ながらキーボードを操作してプログラム」のできるソフトを開発した．

順」に取り出す処理に向いている.

今月のプログラム

最後に説明したもっともシンプルなリングバッファの管理方法を用いて,「起動日時を10組記憶して表示するツール」を作ってみる. この場合のリングバッファの使用目的は,「起動日時データを新しいものから何個かプールする」ということだけである. データ領域がいっぱいになったかというチェックもしないので, いったんデータ領域がフルになるといちばん古いものの上から重ね書きしていく. 表示は最新のものから古いものへと遡って行うことにする (つまりデータの記録と逆順になる. リングバッファを逆にたどっていくわけだ).

記録したデータは, メモリ上に置いておくわけにもいかないので, リングバッファの構造そのままでディスク上のファイルとして作成することにする. ファイル構造は, 図10のような固定フォーマットになる.

動作説明

■メインルーチン

処理のおおまかな流れとしては,

①現在の日時を得てそれを表示する.

②データファイルをオープンし, あればそのデータを読み込んで表示する.

③現在の日時を最新のポインタが示すところへセットする.

④更新されたデータをファイルに書き込んでクローズする.

ということになる. 構造自体は非常に

図7

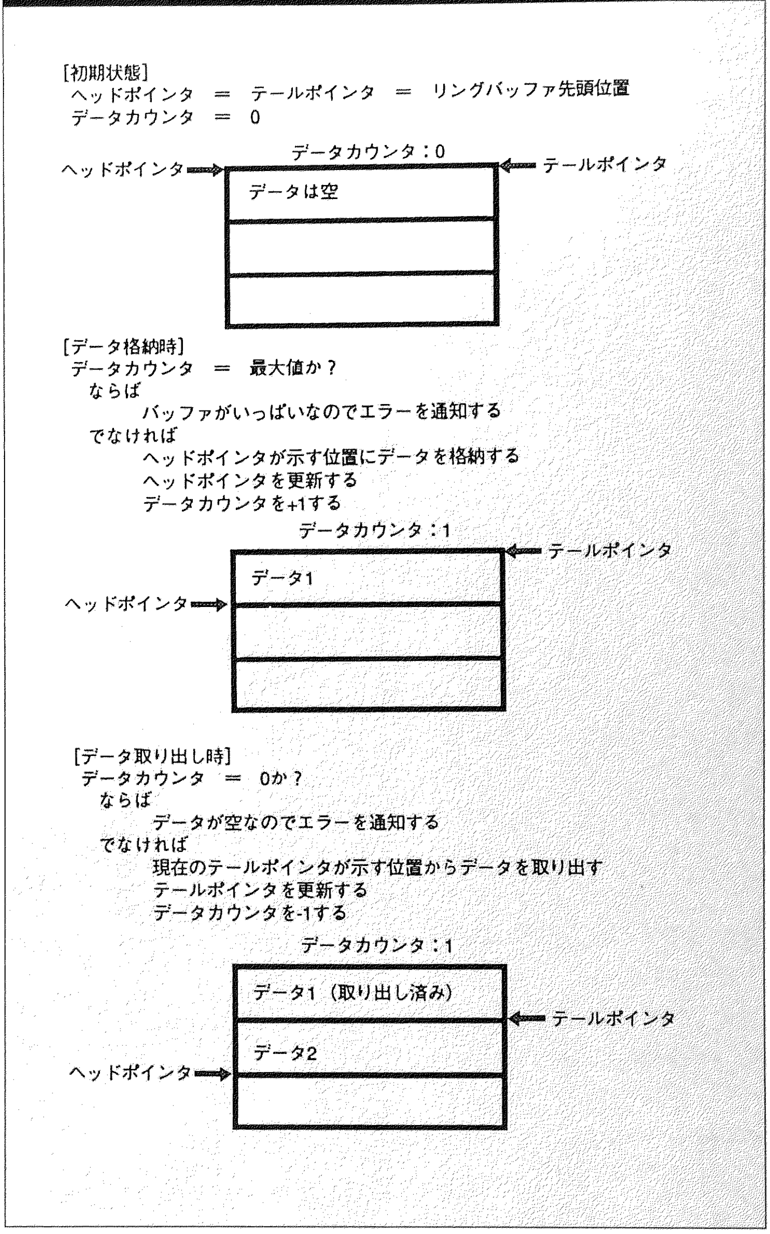
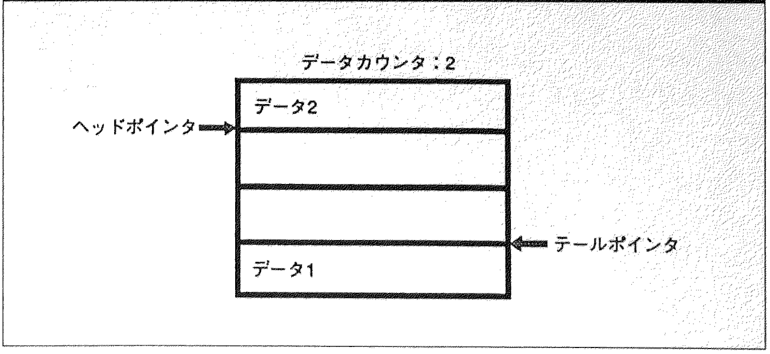


図8



簡単なものなので，ヘタな説明よりリストをご覧になっていただくほうがわかりやすいかもしれない．ここではリングバッファの使い方を具体的に説明することにする．

[バッファ領域]

日時のデータは，1件につき7バイト（「年」が2バイト，「月」「日」「時」「分」「秒」が各1バイト）とし，10個のデータを格納できるように70バイトを確保する．

[ポインタ]

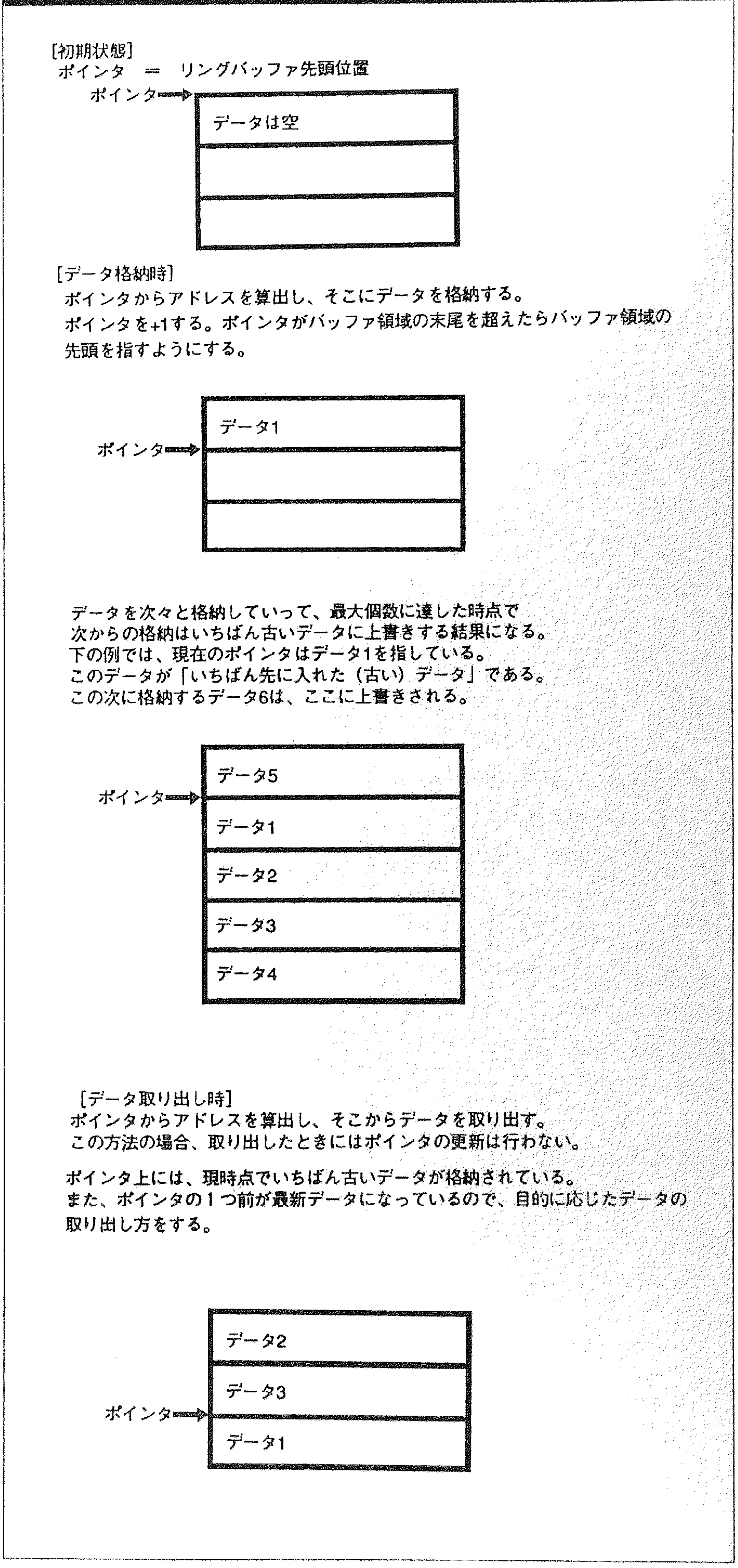
ここで使用するポインタの基本的な考えは，「次にデータを入れる位置を示す」である．ポインタは，0から9まで増加方向へ変化する．9の次は0となる．このポインタを用いてデータのアドレスを以下のように計算する．

アドレス＝バッファ先頭アドレス＋ポインタ＊7

■データファイルの新規作成およびデータ初期化

データファイルの内容を表示する際に，データが10個に満たない場合は，データバッファに入っているものすべてを表示するわけにはいかないので，「どこまでデータが入っているか」の判断が必要となる．データカウンタを用いる手法も考えられるが，このプログラムの場合，いったんデータ数が10に達してしまうとファイルが消去されない限り永劫10のままであるので少々おもしろくない．というわけで，新規作成時にバッファを0でクリアしておき，「0という日時はない」という前提の元に処理を行っている．

図9



ワンラインNEWS
カノープス電子はビデオ画像をパソコンの画面上に高画質で表示するビデオ表示ボード「Super CVI」を発売した．パソコン上でビデオ画像の管理や加工ができる．

■現在の日付・時刻をバッファに設定

ポインタからアドレスを算出してそこへ現在の日時データ(7バイト)を格納し，ポインタを更新する(＋1して9を超えたら0にする)．

■バッファ上の日付・時刻の表示

ポインタが指しているのが「次にデータを格納する位置」であるから，その1つ前が最後に格納されたデータである．表示はここからデータを格納したのと逆の回り方で行っていき，1周して最後はポインタ位置（ここに現在もっとも古いデータが入っている）で終了する．

■ファイルのリード／ライト

唐突にファイルを使いはじめてしまったが，今回はあくまでもデータの待避用にファイルを作って単純に読み，単純に書き込みをするだけである．本来ならばファイル操作はエラー処理をもっときちんとやらなければならないのであるが，今回はファイル操作が主眼ではないというものすごい理由で「何かエラーが発生した」などと省略してしまったことをご了承願いたい．

1つだけ，よくファイル操作でトラブルの原因となりやすい「ファイルポインタの操作」に関して．プログラムの最初のほうでデータファイルを最後までリードしている．このため，更新したデータを再度同じ名前で書き込むためにはファイルポインタを一度最初の位置に戻してやらなければならない．「読む」のと「書く」のとで意識が違うため，混乱しがちだが，リード・ライトともに1つのポインタで管理するので注意が必要である．

■アセンブルとリンクの方法

```
A>MASM TIMERING;
A>LINK TIMERING;
```

※“DOSFUNC.H”は，先月号までのものに追加を行っている．

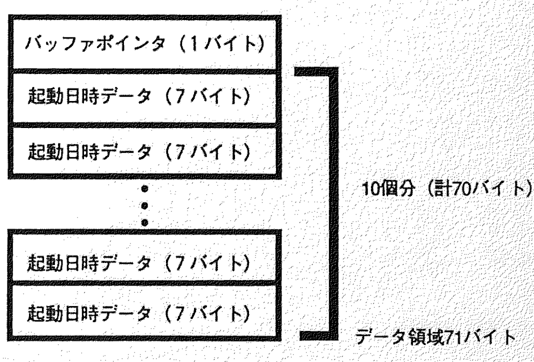
■使い方

```
A>TIMERING<ret>
```

オプションパラメータも何もない．初回の起動では単に現在の日時が表示されるだけである．2回目以降からは記録されている日時が新しいものから順に表示されていく．10回を超えると，古いものから消えていくことを確認していただきたい．

データファイルは，「DATE$$$$.$$$」という71バイトのファイルになる．無条件にカレントディレクトリを探して存在しなければ新規作成するの

図10



リスト2 dosfunc.h

```
;;;;;;;;;;;;;;;;;;;;;
;       dosfunc.h       ;
;;;;;;;;;;;;;;;;;;;;;
;
H_STDOUT        equ 1           ; 標準出力ハンドル
H_ERROR         equ 2           ; 標準エラー出力ハンドル
;
NORMAL_END      equ 0           ; プロセス正常終了コード
;
AT_A            equ 00100000b   ; アーカイブ属性
AT_D            equ 00010000b   ; サブディレクトリ
AT_V            equ 00001000b   ; ボリュームラベル
AT_S            equ 00000100b   ; システムファイル
AT_H            equ 00000010b   ; 隠しファイル
AT_R            equ 00000001b   ; リードオンリーファイル
;
ATTR_ALL        equ AT_A + AT_D + AT_V + AT_S + AT_H + AT_R
;
; === ＩＮＴ－２１Ｈ機能番号 ===
;
DOS_SETDTA      equ 1ah         ; ＤＴＡアドレス設定
DOS_SETVECT     equ 25h         ; 割り込みベクタ値の設定
DOS_GETVECT     equ 35h         ; 割り込みベクタ値の取得
DOS_HWRITE      equ 40h         ; ファイル／デバイスへの出力
DOS_TERMINATE   equ 4ch         ; プロセス終了
DOS_FINDFIRST   equ 4eh         ; 最初のファイルの検索
DOS_FINDNEXT    equ 4fh         ; つぎのファイルの検索
DOS_MALLOC      equ 48h         ; メモリ獲得
;
DOS_PRINTCHR    equ 02h         ; 一文字表示
DOS_PRINTSTR    equ 09h         ; 文字列表示
DOS_GETDATE     equ 2ah         ; 日付の獲得
DOS_GETTIME     equ 2ch         ; 時刻の獲得
DOS_CREATEFILE  equ 3ch         ; ファイルの新規作成
DOS_OPENFILE    equ 3dh         ; ファイルのオープン
DOS_CLOSEFILE   equ 3eh         ; ファイルのクローズ
DOS_READFILE    equ 3fh         ; ファイルの読み込み
DOS_WRITEFILE   equ 40h         ; ファイルの書き込み
DOS_SETFILEPTR  equ 42h         ; ファイルポインタの移動
```

で，PATHを設定して，別のディレクトリからプログラムを起動する場合は注意してほしい．

今月のお勧め

本論とまるで関係のないところに「お勧め」があるのは少々気がひけるが，アセンブラでツールを書くときに意外と面倒な「数値の10進表示」をご利用いただきたい．4月号でも数値の10進変換ルーチンをお勧めしているが，こちらはあれとは違ってテーブルを使ったロジックである．同じ結果を得るための方法は1つではないという見本．

今回は，「年」の表示だったので1000の位までしか考慮していないが，テーブルに「10000」を加えるだけで，汎用の2バイトデータ10進表示ルーチンとなる．お試しあれ．

今月の反省

今月のプログラムは，サンプルプログラムの色が濃く，ツールとしての実用性はあまりなくなってしまった．書いてから使いみちを考えるのは邪道ではあるが，起動日時の記録を利用した「運用記録プログラム」や，前回起動日

リスト3 timering.asm

```
TITLE   Date/Time ring

PAGE        60,128

.xlist
include     dosfunc.h
.list

_TEXT       SEGMENT WORD PUBLIC 'CODE'
_TEXT       ENDS

_DATA       SEGMENT WORD PUBLIC 'DATA'
_DATA       ENDS

_STACK      SEGMENT STACK
_STACK      ENDS

DGROUP      GROUP   _DATA,_STACK
            ASSUME  DS:DGROUP, SS:DGROUP

;
; ============= 定数宣言 ======================
;
MAX_DATA    equ 10                      ; リングバッファの最大個数

date_struc      struc                   ; データ構造の型紙
year    dw      0
month   db      0
date    db      0
hour    db      0
mini    db      0
seco    db      0
date_struc      ends
;
; ============================================
;
_TEXT       SEGMENT
        ASSUME CS:_TEXT

;--------------- ●バッファの初期化及びデータファイル新規作成
create_new      proc
        mov     dx,offset data_file_name
        mov     cx,0
        mov     ah,DOS_CREATEFILE
        int     21h                     ; ファイルの新規作成
        mov     bp,ax                   ;

        mov     di,offset buff_point
        mov     cx,buffer_length
        xor     al,al
        rep     stosb                   ; データバッファを0クリアする
        ret
create_new      endp

;--------------- ●データ格納（現在の日付・時刻をバッファに設定）
set_current     proc
        mov     bl,[buff_point]         ; 現在のポインタ位置に書き込む
        mov     al,current_length
        mul     bl
        mov     di,offset date_buff
        add     di,ax                   ; ポインタからアドレスの計算
        mov     cx,current_length
        mov     si,offset current_date
        rep     movsb                   ; データをコピーする
        inc     bl                      ; ポインタの更新処理
        cmp     bl,MAX_DATA             ; ポインタが最大値まで来たか？
        jc      next_set
        xor     bl,bl                   ; ならば先頭へ戻る
next_set:
        mov     [buff_point],bl         ; 新しいポインタをセットする
        ret
set_current     endp

;--------------- ●データ取り出し
disp_data       proc
        mov     dx,offset data_mes
        mov     ah,DOS_PRINTSTR
        int     21h                     ; 見出しの表示
        xor     cx,cx
        mov     cl,[buff_point]
disp_data_loop:
        or      cl,cl                   ; ポインタが最少になっていれば
        jnz     update_point
        mov     cl,MAX_DATA             ; 最大へ移動しておいて
update_point:
        dec     cl                      ; いちばん新しいデータから始める
        push    cx
        mov     al,cl
        mov     bl,current_length
        mul     bl                      ; ポインタからアドレスを計算
        mov     bx,offset date_buff
        add     bx,ax
        cmp     word ptr [bx],0         ; データが入っているかどうか？
        jz      skip_disp               ; 入っていなければスキップする
        call    disp_date               ; データ位置から表示
skip_disp:
        pop     cx
        cmp     cl,[buff_point]         ; 終了判定
        jz      disp_data_end
        jmp     disp_data_loop
disp_data_end:
        ret
disp_data       endp

;--------------- 現在の日付・時刻を得て表示する
disp_current    proc
        mov     dx,offset current_mes
        mov     ah,DOS_PRINTSTR
        int     21h                     ; 見出しの表示
        mov     ah,DOS_GETDATE
        int     21h                     ; 「日付を得る」

        mov     bx,offset current_date  ; ワークバッファへ保存
        mov     [bx.year],cx
        mov     [bx.month],dh
        mov     [bx.date],dl

        mov     ah,DOS_GETTIME          ; 「時刻を得る」
        int     21h

        mov     [bx.hour],ch            ; ワークバッファへ保存
        mov     [bx.mini],cl
        mov     [bx.seco],dh

        call    disp_date               ; ワークバッファから表示する
        ret
disp_current    endp

;--------------- 日付・時刻の表示サブルーチン
disp_date       proc
        push    bx                      ; ＢＸを先頭とする構造体を処理
        mov     ax,[bx.year]            ; 「年」の表示（これだけ４桁）
        call    disp_1word
        call    disp_sep1               ; 区切りは「/」
        mov     al,[bx.month]           ; 「月」の表示
        call    disp_1byte
        call    disp_sep1               ; 区切りは「/」
        mov     al,[bx.date]            ; 「年」の表示
        call    disp_1byte
        call    disp_sep3               ; 区切りは「 」

        mov     cx,3                    ; 「時・分・秒」の表示
        add     bx,hour
        jmp     short time_loop_ent     ; 最後の一回は区切り表示なし
time_disp_loop:
        call    disp_sep2               ; 区切りは「:」
time_loop_ent:
        mov     al,[bx]
        inc     bx
        call    disp_1byte
        loop    time_disp_loop
time_disp_end:
        mov     dl,0dh                  ; 一行表示して改行する
        call    disp_1char
        mov     dl,0ah
        call    disp_1char
        pop bx
        ret
disp_date       endp
```


ワンラインNEWS
マイクロヘリオスは米サンマイクロシステムズのワークステーションと完全互換品を発売した．低価格でUNIXの学習用にと個人需要にも期待する．

時と現在のカレンダを比較してのカレンダ設定の整合性チェックツールなどに応用していただければ幸いである．

ところで，私事にわたって恐縮ではあるが，我がジャステックラボも「ソフトハウスだといい張っている」という状態から，まがりなりにも会社の体裁をとることができるようになった．事務所も借りたし，社員も入った．別に威張るような話ではないが，コーヒーメーカーも買った．これらの資金の一部に，本連載の原稿料が充てられたことはいうまでもなく，会社設立にこぎつけたのは読者のご支持と編集者の忍耐のおかげである．心からお礼申し上げる．

なお，今月のプログラムは筆者のスケジュールの都合で社員の１人が担当した．プログラムスタイルが違うのはそのせいである．どちらがうまいかなどということは詮索しないでもよろしい．する必要はない．しないほうがいい．しないでってば．

```
;--------------- 1ワードデータを10進表示（最大4桁）
; AX <-- データ
disp_1word      proc
        push    bx
        mov     cx,3                    ; '1000','100','10'と順に変換
        mov     si,offset num_table
word_loop:
        xor     dx,dx
        mov     bx,[si]
        div     bx                      ; 各々の桁で割って商を表示して
        push    dx
        add     al,'0'
        mov     dl,al
        call    disp_1char
        pop     dx
        mov     ax,dx                   ; その余りで次の処理へ
        inc     si
        inc     si
        loop    word_loop
        add     dl,'0'
        call    disp_1char              ; 最後に1の位を表示して終わり
        pop     bx
        ret
disp_1word      endp
;--------------- 1バイトデータを10進表示（最大2桁）
; AL <-- データ
disp_1byte      proc
        push    bx
        mov     bl,al
        xor     dl,dl
ten_count:                              ; 「10の位」をカウントする
        sub     bl,10
        jc      end_ten_count
        inc     dl
        jmp     short ten_count
end_ten_count:
        add     bl,10
        add     dl,'0'
        call    disp_1char              ; 10の位の表示
        add     bl,'0'
        mov     dl,bl
        call    disp_1char              ; 1の位の表示
        pop     bx
        ret
;--------------- 区切り文字の表示
disp_sep1:
        mov     dl,'/'
        jmp     short disp_sep_common
disp_sep2:
        mov     dl,':'
        jmp     short disp_sep_common
disp_sep3:
        mov     dl,' '
disp_sep_common:
        call    disp_1char
        ret
disp_1char:
        mov     ah,DOS_PRINTCHR
        int     21h
        ret
disp_1byte      endp
;--------------- データファイルをリードしてバッファに満たす
get_data        proc
        mov     dx,offset data_file_name
        mov     al,2
        mov     ah,DOS_OPENFILE         ; データファイルをオープン
        int     21h
        jc      open_fail               ; エラーならば読まずにリターン
        mov     bp,ax
        mov     bx,ax
        mov     dx,offset buff_point
        mov     cx,buffer_length
        mov     ah,DOS_READFILE         ; データのリード
        int     21h
open_fail:
        ret
get_data        endp
;--------------- バッファの内容をデータファイルに書き込んでクローズ
set_data        proc
        mov     bx,bp
        xor     al,al                   ; ファイルポインタを先頭へ移動
        xor     cx,cx
        xor     dx,dx
        mov     ah,DOS_SETFILEPTR
        int     21h
        mov     dx,offset buff_point
        mov     cx,buffer_length
        mov     ah,DOS_WRITEFILE        ; ファイルへの書き込み
        int     21h
        jc      write_fail
        mov     ah,DOS_CLOSEFILE        ; データファイルをクローズ
        int     21h
write_fail:
        ret
set_data        endp
;----------------------------------------------
main    proc
        mov     ax,DGROUP               ; ES=DS=ﾃﾞｰﾀｾｸﾞﾒﾝﾄ
        mov     ds,ax
        mov     es,ax
        call    disp_current            ; 現在の日付・日時を表示
        call    get_data                ; データファイルのリード
        jnc     data_exist
        call    create_new              ; もし無ければ新しく作る
        jc      error_exit
        jmp     short data_not_exist
data_exist:
        call    disp_data               ; データ内容の表示
data_not_exist:
        call    set_current             ; 最新のデータをバッファに設定
        call    set_data                ; バッファ内容を書き出してクローズ
        jc      error_exit
        mov     ah,DOS_TERMINATE
        int     21h
error_exit:
        mov     dx,offset error_mes
        mov     ah,DOS_PRINTSTR
        int     21h
        mov     ah,DOS_TERMINATE
        int     21h
main    endp
_TEXT   ENDS
_STACK  SEGMENT STACK
                dw      128     dup (?)
_STACK  ENDS
_DATA   SEGMENT WORD PUBLIC 'DATA'
data_file_name  db      'date$$$$.$$$',0
current_mes     db      '=== 今回の起動日時 ===',0dh,0ah,'$'
data_mes        db      '------ 起動記録 ------',0dh,0ah,'$'
error_mes       db      'ファイルアクセスでエラー発生...$'
buff_point      db      0
date_buff       date_struc  10 dup (<>)
buffer_length   equ     $ - offset buff_point
current_date    date_struc <>
current_length  equ     $ - offset current_date
num_table       dw      1000
                dw      100
                dw      10
_DATA   ENDS
        END     main
```

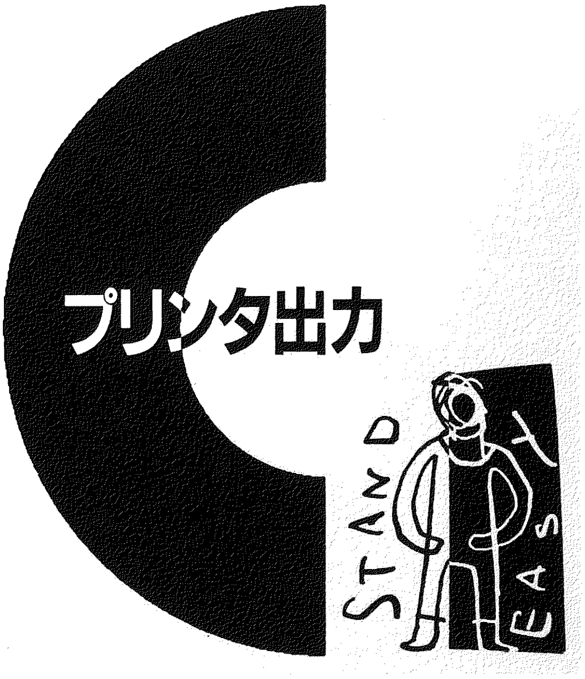
ワンラインNEWS
YHPは積水化学工業からRISC型サーバー「HP9000シリーズ800マルチユーザーUNIXシステム」を軸にしたコンピュータシステムを受注し，納品した．

実践C塾

Illustration:
Masami Nozue

基礎テクニック篇　第5回

プリンタ出力



二代目塾頭　大野元久

プリンタへデータを出力するために，MS-DOS上のC言語処理系の多くがstdprnというストリームを備えている．しかし，PC-9801のMS-DOSでは，このstdprnを使用するためにPRINT.SYSを前もって組み込んでおかなければならないなどの制限がある．そこで，これらの制限を受けないプリンタBIOSを利用したプリンタ出力モジュールを設計する．

プリンタBIOSを使った出力モジュール

プリンタへデータを出力する場合，何を使えばよいでしょうか．MS-DOS用のCの処理系は，たいていstdprnというプリンタ用のストリームを用意しています．たとえば，fprintf (stdprn, "Hello, world!¥r¥n")；とすればプリンタに"Hello, world!"という文字列が出力されます．通常，stdprnはMS-DOSの標準プリンタデバイスに連結されていますから，これを使えばどのMS-DOSマシンでもプリンタ出力ができるはずです．ところが，最近のPC-9801用のMS-DOSではPRINT.SYSを組み込まないと，標準プリンタデバイスが使用できません．自分で使うだけならPRINT.SYSを組み込んでしまえばよいのですが，ほかの人にも使ってもらおうというアプリケーションなら，できればそうした制限はなくしたいものです．それに，PRINT.SYSを組み込んでいるときは，少々めんどうな手続きを踏まないとビットイメージを出力することができません．こうした制約は，グラフィックス画面のハードコピーをとりたい場合に障害となります．というわけで，こうしためんどうな手間を考えるくらいなら，プリンタBIOSを使ってしまおうということで，プリンタBIOSを使ったプリンタ出力モジュールを作成することにしました．なお，今回はノーマルモードだけを対象とし，ハイレゾリューションモードは考慮しません．また，プリンタはPC-PRシリーズかNMシリーズの24ドットプリンタのみを対象とし，EPSONのESC/P仕様は考慮しません．

ノーマルモードのプリンタBIOSは，表1のような仕様になっています．プリンタBIOSは，AHが30H以外の場合，正常終了するとAHの最下位ビットを1にし，そうでなければ0にします．ビット1にはタイムアウト情報が返されますが，通常は正常に終了したかどうかさえ判定できれば問題ないでしょう．今回は，正常に出力できるまで，同じ文字コードを出力し続けることにします．

1バイト出力

プリンタへ1バイトのデータを出力するための関数を，リスト1，リスト2に示します．lprint.hには，このあとで作成する関数のための定義やプロトタイプ宣言も含まれています．

lputc.cで定義されている関数の仕様は表2のとおりです．

これらの関数の中で，_lstatはプリンタ出力の前にプリンタの状態を確認

表1　プリンタBIOS
割り込みベクタ　INT 1AH

AHの値	意味
10H	プリンタインターフェースの初期化
11H	プリンタへ1バイト(AL)を出力
12H	プリンタの状態を取得
30H	文字列をプリンタに出力（ES：BXからCXバイト）

表2

_lputcraw	
機能	プリンタへ直接1バイトを出力する．
形式	int _lputcraw (int c)；
説明	プリンタBIOSを使って1バイト出力します．正常終了の場合は出力した文字コードを返し，エラーの場合はEOFを返します．

_lstat	
機能	プリンタの状態を返す．
形式	int _lstat (void)；
説明	プリンタBIOSを使って，プリンタの状態を取得します．出力可能状態であれば1を，そうでなければ0を返します．

_lputc	
機能	プリンタが出力可能になるまで待って，1バイト出力する．
形式	int _lputc (int c)；
説明	基本的に_lputcrawと同じですが，プリンタが出力可能状態になるまで待ってから1バイト出力します．

_lputs	
機能	プリンタへの文字列出力
形式	int lputs (const char *str)；
説明	_lputcを使ってASCIZ文字列を出力します．

ワンラインNEWS
石川島播磨重工業は液晶パネルを構成する最小単位である画素の欠陥とその種類を生産過程で検出する検査装置を開発した．

するために使います．たとえば，リスト3のようにすれば，印字前にプリンタの準備ができているかどうか確認できます．しかし，PC-9801の場合，プリンタの電源が入っていないときでも，プリンタが印字可能状態であると判定されてしまいます．この場合は，プリンタにディセレクト(13H)コードを出力して，プリンタがディセレクト（印字不能）状態になるかどうか確認することができます．ディセレクトコードを出力しても，プリンタの電源が入っていなければlstat () は1を返します．電源が入っていれば，プリンタは印字不能状態になりlstat () は0を返します．

しかし，ここで示したBIOSの機能では，印字不能状態から印字可能状態に戻すセレクト(11H)コードを出力することができません．セレクトコードの出力は，直接I/Oポートを操作することになります．I/Oポートを操作することは，それほど難しいことではありません．しかし，プリンタによってプリンタ側から印字不能状態にした場合，プリンタ本体のスイッチでは，印字可能状態に変更できないものがあります．また，プリンタバッファを使用している場合は，効力がなくなるばかりか，永久に印字不能に陥ってしまう可能性もあります．

リスト3 （プリンタ状態の検査）

```
while (_lstat() == 0) {
    printf("プリンタの準備ができたらキーを押してください: ");
    getch();
    printf("¥n");
}
/* 印字処理 */
```

リスト1 （lputc.c）

```
/*
 *  lputc.c: プリンタ出力モジュール（ＢＩＯＳ操作）
 *       written by Motohisa Ohno
 */
#include <dos.h>
#include "lprint.h"

#define EOF (-1)

/* タブカラム数 */
int ltabstop = 8;
/* 現在の桁位置 */
int lcolumn = 0;

/*
 *  プリンタ BIOS を使って1バイト出力する
 */
int _lputcraw(int c)
{
    union REGS regs;

    regs.h.ah = 0x11;
    regs.h.al = (unsigned char)c;
    int86(PBIOS, &regs, &regs);
    return (regs.h.ah & 1) != 0 ? c : EOF;
}

/*
 *  プリンタステータスを取得する
 *       戻り値＝0  使用不可
 *             ＝1  使用可
 */
int _lstat(void)
{
    union REGS regs;

    regs.h.ah = 0x12;
    int86(PBIOS, &regs, &regs);
    return (regs.h.ah & 1) != 0;
}

/*
 *  プリンタステータスが有効になるのを待って1バイト出力
 */
int _lputc(int c)
{
    while (_lstat() == 0)
        ;
    return _lputcraw(c);
}

/*
 *  プリンタへ文字列出力
 */
int _lputs(const char *str)
{
    while (*str) {
        _lputc((unsigned char)*str);
        ++str;
    }
    return 0;
}
```

リスト2 （lprint.h）

```
/*
 *  各種シンボルの定義
 */
#define CRTBIOS 0x18     /* CRT-BIOS の割り込みベクタ */
#define PBIOS 0x1a       /* プリンタ BIOS の割り込みベクタ */

#define ROWMAX 25        /* 画面の最大行数 */
#define COLMAX 80        /* 画面の最大桁数 */

typedef unsigned char BYTE;      /* バイト型(unsigned char) */
typedef unsigned short WORD;     /* ワード型(unsigned short) */

/*  タブカラム数  */
extern int ltabstop;
/*  現在の桁位置  */
extern int lcolumn;

/*
 *  1バイト出力モジュール
 */
int _lputcraw(int c);
int _lstat(void);
int _lputc(int c);
int _lputs(const char *str);

/*
 *  文字、文字列出力モジュール
 */
int lputc(int c);
int lputs(const char *str);
int lprintf(const char *format, ...);

/*
 *  文字、文字列出力モジュール（縮小印字版）
 */
int xputc(int c);
int xputs(const char *str);
int xprintf(const char *format, ...);

/*
 *  ハードコピーモードのための列挙型シンボル
 */
enum copymode_t {
    HCOPY_COPY = 0,              /* 最後のモードでコピーする */
    HCOPY_BLUE = 0x0001,         /* BLUE プレーンのコピーをとる */
    HCOPY_RED = 0x0002,          /* RED プレーンのコピーをとる */
    HCOPY_GREEN = 0x0004,        /* GREEN プレーンのコピーをとる */
    HCOPY_TEXT = 0x0010,         /* テキスト画面のコピーをとる */
    HCOPY_GRAPH = HCOPY_BLUE | HCOPY_RED | HCOPY_GREEN,
    HCOPY_ALL = HCOPY_TEXT | HCOPY_GRAPH,

    HCOPY_PR = 0x0100,           /* PC-PR 系のプリンタを使用する */
    HCOPY_NM = 0x0200,           /* NM 系のプリンタを使用する */

    HCOPY_SETMODE = (int)0x8000, /* 実際にコピーはしない */
};

/*
 *  ハードコピーモジュール
 */
void hcopypart(int mode, int left, int top, int right, int bottom);
void hcopy(int mode);
```


ワンラインNEWS
日本テクニカル工業は日本IBM製のパソコン「PS/55シリーズ」に対応したキーボード「PSエントリーⅠ」を発売．大きさをフルサイズキーボードの約1/4としている．

書式つき出力

漢字に対応した文字出力関数をリスト4に示します．

lprint.cで定義されている関数の仕様は表3のとおりです．lprint.cを使うときは，lputc.cとともにコンパイル・リンクしなければなりません．

setpmodeは，プリンタの印字モードを漢字モードとANKモードのどちらにするかを設定する関数です．プリンタをどちらのモードに変更したかを記憶しておくことで，モードの設定が要求されても，以前のモードと同じであれば，プリンタにはモード設定のエスケープシーケンスを出力しません．

lputcは，漢字コードを処理する文字出力関数の基本となる関数です．プリンタBIOSは，指定された文字コードを1バイト出力するだけですから，そのままでは漢字を出力することができません．PC-9801用のプリンタでは，漢字を出力する場合は，ESC 'K'で漢字モードに変更したあとで，漢字のJISコードを上位バイト，下位バイトの順で出力します．ANK（半角）モードに戻すときは，ESC 'H'を出力します．

lputcが，シフトJISコードの1バイト目を受けとると，そのコードを内部変数（prknj）に保留し，次に渡された文字コードは無条件で漢字の2バイト目と認識されます．これをjmstojisでJISコードに変換した後，プリンタを漢字モードにして上位バイト，下位バイトの順で出力します．この関数は，半角文字が渡されたときも，漢字モードにおける半角文字が使用されます．こ

リスト4 (lprint.c)

```
/*
 *  lprint.c: プリンタ出力モジュール (漢字対応版)
 *      written by Motohisa Ohno
 */
#include <stdio.h>
#include <stdarg.h>
#include <ctype.h>
#include <jctype.h>
#include <string.h>
#include <jstring.h>
#include "lprint.h"

/*
 *  出力中のモード
 */
static enum pmode_t {
    PMODE_ANK,      /* ANK 出力中 */
    PMODE_KANJI,    /* 漢字出力中 */
} pmode = PMODE_ANK;

static WORD prknj = 0;  /* 漢字フラグ */

/*
 *  プリンタの出力モードを指定
 */
static void setpmode(enum pmode_t newmode)
{
    if (pmode == newmode)
        return;
    switch (newmode) {
    case PMODE_ANK:
         _lputc('¥033');
         _lputc('H');
        break;
    case PMODE_KANJI:
         _lputc('¥033');
         _lputc('K');
        break;
    }
    pmode = newmode;
}

/*
 *  プリンタに1文字出力
 */
int lputc(int c)
{
    if (prknj) {
        setpmode(PMODE_KANJI);
        prknj = jmstojis((prknj << 8) | (BYTE)c);
         _lputc((unsigned char)(prknj >> 8));
         _lputc((unsigned char)prknj);
        lcolumn += 2;
        prknj = 0;
        return c;
    }
    if (iskanji((unsigned char)c)) {
        prknj = c;
        return c;
    }
    if (c == '¥t') {
        do {
             lputc(' ');
        } while (lcolumn % ltabstop);
        return c;
    }
    if (iscntrl((unsigned char)c)) {
        setpmode(PMODE_ANK);
        if (c == '¥r' || c == '¥n') lcolumn = 0;
        if (c == '¥n') _lputc('¥r');
        if (c != '¥r') _lputc(c);
        return c;
    }
    setpmode(PMODE_KANJI);
     _lputc(0);
     _lputc(c);
    ++lcolumn;
    return c;
}

/*
 *  プリンタへ文字列出力
 */
int lputs(const char *str)
{
    while (*str) {
        lputc((unsigned char)*str);
        ++str;
    }
    setpmode(PMODE_ANK);
    return 0;
}

#if defined(__TURBOC__) && (defined(__TINY__) || defined(__SMALL__))

typedef unsigned pascal near putnF (const void *ptr, unsigned n, void *fp);
int pascal near __vprinter(putnF near *__putter, void *__outP,
                           const char *__formatP, void _ss *__argP);

unsigned pascal near __lputn(const void *ptr, unsigned n, void *fp)
{
    while (n--)
        lputc(*((char *)ptr)++);
    return n;
}

/*
 *  プリンタへ書式付き文字列を出力
 */
int lprintf(const char *format, ...)
{
    return __vprinter((putnF *)__lputn, NULL, format, (void _ss *)_va_ptr);
    /*setpmode(PMODE_ANK);*/
}

#else /* MS-C or Turbo C++'s large code,data model */

/*
 *  プリンタへ書式付き文字列を出力
 */
int lprintf(const char *format, ...)
{
    char buf[256];
    va_list ap;
    int i, n;

    va_start(ap, format);
    n = vsprintf(buf, format, ap);
    for (i = 0; i < n; i++) lputc(buf[i]);
    va_end(ap);
    return n;
}

#endif
```

ワンラインNEWS
日本電算機は小型・低価格のカラーXウィンドウ端末「SuperXst.L」を発売した．1670万色中256色の同時発色が可能．価格はモニター込みで60万円を切っている．

の文字コードは，1バイト目が0で2バイト目に文字コードを指定します．これにより，ANKと漢字の比率が常に1:2となります．

改行文字は，復帰／改行のペアに変更されます．逆に，復帰コードが渡されても無視されます．制御文字が送られると，プリンタの印字モードはANKモードに戻されますが，これは必ずしも必要な処理ではありません．改行文字を受け取ったあとに，別の印字処理が要求される可能性に配慮したためです．

lputsは，文字コード0があらわれるまで，文字列を1文字ずつlputcに渡します．lputsでは，プリンタのエスケープシーケンスを出力できないことに注意してください．lputcは，ANKを漢字モードの半角文字に変換してしまいますから，ESC _A'を出力してもプリンタには1BH，00H，41Hが出力されてしまい，正しく処理されません．エスケープシーケンスは，lputc.cに用意されている_lputcを組み合わせるか，_lputs関数を使ってください．

lprintfは，プリンタへの書式つき出力を実現するものです．Turbo C++のタイニィモデルとスモールモデルでは，Turbo C++の非公開関数__vprinterを呼び出すことで，汎用性の高い書式つき出力関数を実現しています．この非公開関数は，near呼び出しでなければならないので，ラージモデルからは呼び出せません．ちなみに，Turbo C++のランタイムライブラリは，-zオプションを使って__vprinterとprintfなどのライブラリ関数を同じセグメントに配置することで，near呼び出しを実現し高速化をはかっています．

MS-Cでは，書式を解釈するための内部関数が流用できるものではなさそうだったので，Turbo C++のラージモデルと同様vsprintfを使って，いったん書式を解釈した結果を文字バッファに展開し，それを出力するようにしています．このため，一度に256文字以上の文字列を出力できません．

この印字モジュールを使ったプログラム例をリスト5に示します．lcat.cは，lprint.cとlputc.cとともにコンパイル・リンクしてください．このプログラムは，標準入力（あるいはコマンドラインで指定されたファイル）の内容を標準出力でなく，プリンタへ出力するものです．たとえば，lcat lprint.cとすればlprint.cの内容がプリンタに出力されます．

表3

lputc	
機能	プリンタに1文字出力する．
形式	int lputc (int c)；
説明	プリンタに1文字出力します．漢字を出力する場合は，漢字のシフトJISコードを上位バイト(1バイト目)，下位バイト(2バイト目)の順で与え，連続して呼び出します．漢字の2バイト目が指定されなかった場合，その後の動作は不定となります．
lputs	
機能	プリンタに文字列を出力する．
形式	int lputs (char ＊str)；
説明	関数lputcを使って，ASCIZ文字列を出力します．関数の最後でプリンタの印字モードをANKモードに戻しています．
lprintf	
機能	プリンタへの書式つき出力．
形式	int lprintf (const char ＊format, ...)；
説明	関数lputcを使って，プリンタへの書式つき出力を行います．

リスト5 (lcat.c)

```
/*
 *  lcat.c: プリンタモジュールを使ったcat
 *       written by Motohisa Ohno
 */
#include <stdio.h>
#include <stdlib.h>
#include "lprint.h"

FILE *efopen(const char *file, const char *mode)
{
    FILE *fp;

    fp = fopen(file, mode);
    if (fp == NULL) {
        fprintf(stderr, "can't open %s(%s)¥n", file, mode);
        exit(1);
    }
    return fp;
}

void main(int argc, char **argv)
{
    char buf[BUFSIZ];
    FILE *fp;

    if (argc == 1) {
        while (fgets(buf, sizeof buf, stdin))
            lputs(buf);
        return;
    }
    while (--argc) {
        fp = efopen(*++argv, "r");
        while (fgets(buf, sizeof buf, fp))
            lputs(buf);
        fclose(fp);
    }
}
```


ワンラインNEWS
情報処理振興事業会はコンピュータウイルスの情報を収集・提供するためのパソコンネットワークを開設．電話番号は03-3459-8944で月曜の午前を除き24時間稼働

縮小印字

プリンタを有効に使うため，文字を縮小して印字したいということがあります．リスト6には，PC-9801の内蔵フォント(16ドット)を使って，通常の24ドットフォントよりもひとまわり小さい文字で出力する関数が用意されています．

xprint.cで定義されている関数の仕様は表4のとおりです．xprint.cを使うときは，lputc.cとともにコンパイル・リンクしなければなりません．

xprint.cは，前々回まで使われていたturboc.hがインクルードされていますが，今回はfarポインタ用のマクロの定義を変更するためだけに使われていますので，この内容はリスト7のようなものでかまいません．

xprint.cの先頭で定義されているpc98getfontは，Turbo C++で用意されている文字フォントを取得するためのライブラリ関数のサブセットです．Turbo C++では，文字コードがシフトJISでもJISでも受け付けられるようになっていますが，この関数はJISコードのみが有効です．

さて，このモジュールの中でもっとも基本となるのがxputc関数です．漢字コードの扱いはlputcと同じです．シフトJISコードが渡されると，それをJISコードに変換します．ANKの16ドットの文字フォントは，文字コードの上位ワードを80Hにしなければならないので8000Hを加算しています．この文字コードをpc98getfontに与えて，文字フォントを取得します．

文字フォントは，最大34バイトのバッファに得られますが，先頭から2バイト目(fbuf [1])には文字の幅が8ド

リスト6 (xprint.c)

```
/*
 *  xprint.c: プリンタ出力モジュール（縮小印字版）
 *      written by Motohisa Ohno
 */
#include <stdio.h>
#include <stdarg.h>
#include <ctype.h>
#include <jctype.h>
#include <string.h>
#include <jstring.h>
#include <dos.h>
#include "turboc.h"
#include "lprint.h"

#if !defined(__TURBOC__)
/*
 *  MS-C 用のフォントを取得するための関数 (Turbo C++ のサブセット)
 */
static void pc98getfont(unsigned code, void *buffer)
{
    union REGS regs;

    regs.h.ah = 0x14;
    regs.x.bx = FP_SEG(buffer);
    regs.x.cx = FP_OFF(buffer);
    regs.x.dx = code;
    int86(CRTBIOS, &regs, &regs);
}
#endif

static WORD prknj = 0;

/*
 *  プリンタに1文字出力
 */
int xputc(int c)
{
    BYTE fbuf[34];
    BYTE image[24];
    BYTE mask;
    WORD pat;
    int i, j, n, w;
    WORD code;

    if (prknj) {
        code = jmstojis((prknj << 8) | (unsigned char)c);
        prknj = 0;
    } else if (iskanji((unsigned char)c)) {
        prknj = c;
        return c;
    } else if (c == '¥t') {
        do {
            xputc(' ');
        } while (lcolumn % ltabstop);
        return c;
    } else if (iscntrl((unsigned char)c)) {
        if (c == '¥r' || c == '¥n') lcolumn = 0;
        if (c == '¥n') {
            _lputc('¥r');
            _lputs("¥033T16");
        }
        if (c != '¥r') _lputc(c);
        if (c == '¥n') _lputs("¥033A");
        return c;
    } else {
        code = (BYTE)c + 0x8000;
    }
    pc98getfont(code, fbuf);
    w = fbuf[1];
    for (n = 0; n < w; n++) {
        for (i = 0, mask = 0x80; i < 8; i++, mask >>= 1) {
            for (j = 0, pat = 0; j < 16; j++) {
                pat >>= 1;
                if (fbuf[2+n+j*w] & mask) pat |= 0x8000;
            }
            image[i*3] = (BYTE)pat;
            image[i*3+1] = (BYTE)(pat >> 8);
            image[i*3+2] = 0;
        }
        _lputs("¥033J0008");
        for (i = 0; i < 8*3; i++) _lputc(image[i]);
        ++lcolumn;
    }
    return c;
}

int xputs(const char *str)
{
    while (*str) {
        xputc((unsigned char)*str);
        ++str;
    }
    return 0;
}

#if defined(__TURBOC__) && (defined(__TINY__) || defined(__SMALL__))

typedef unsigned pascal near putnF (const void *ptr, unsigned n, void *fp);
int pascal near __vprinter(putnF near *__putter, void *__outP,
                           const char *__formatP, void _ss *__argP);

unsigned pascal near __xputn(const void *ptr, unsigned n, void *fp)
{
    while (n--)
        xputc(*((char *)ptr)++);
    return n;
}

int xprintf(const char *format, ...)
{
    return __vprinter((putnF *)__xputn, NULL, format, (void _ss *)_va_ptr);
}

#else /* MS-C or Turbo C++'s large code,data model */

int xprintf(const char *format, ...)
{
    char buf[256];
    va_list ap;
    int i, n;

    va_start(ap, format);
    n = vsprintf(buf, format, ap);
    for (i = 0; i < n; i++) xputc(buf[i]);
    va_end(ap);
    return n;
}

#endif
```

ットか16ドットかによって１か２が返されます．このため，文字コードを調べなくても，正しく２バイト半角文字を処理できます．プリンタのビットイメージ出力では，ビットイメージを縦方向を基準に出力しなければならないのですが，文字フォントは横方向に並んでいます．したがって，文字フォントをビットイメージ出力として適当なデータに変換します．データを変換したら，ESC _J0008'というエスケープシーケンスを使って，横８ドット分のビットイメージ(半角１文字に相当)を出力します．

制御文字については，改行文字を復帰／改行のペアに変換し，復帰文字を無視する以外は，そのまま出力します．ただし，改行する前後で改行幅を24ドットに変更しています．NMプリンタの場合は，_lputs ("¥033T16") ;という行を_lputs ("¥033T18") ;に変更しなければなりません．

xputs，xprintfは，それぞれxputcを使ってASCIZ文字列の出力や書式つき出力を実現するものです(xprintfの手法については，前述のlprintfを参照してください)．

この縮小印字を使ったプログラム例をリスト８に示します．xcat. cは，xprint. cとlputc. cとともにコンパイル・リンクしてください．このプログラムは，標準入力（あるいはコマンドラインで指定されたファイル）の内容を標準出力でなくプリンタへ出力するものです．たとえば，xcat xprint. cとすれば，xprint. cの内容が縮小文字でプリンタに出力されます．

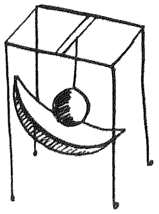


表4

xputc

機能	プリンタに１文字出力する（縮小印字版）．
形式	int xputc (int c) ;
説明	プリンタに16ドットフォントを使って１文字出力します．漢字を出力する場合は，漢字のシフトJISコードを上位バイト(１バイト目)，下位バイト(２バイト目)の順で与え，連続して呼び出します．漢字の２バイト目が指定されなかった場合，その後の動作は不定となります．

xputs

機能	プリンタに文字列を出力する（縮小印字版）．
形式	int xputs (char ＊str) ;
説明	関数xputcを使って，ASCIZ文字列を出力します．

xprintf

機能	プリンタへの書式つき出力．
形式	int xprintf (const char ＊format, ...) ;
説明	関数xputcを使って，プリンタへの書式つき出力を行います．

リスト７ (turboc.h)

```
/*
 *  Microsoft C のマクロ、構造体、関数の定義を Turbo C++ 形式に変更する。
 */
#ifndef __TURBOC_H
#define __TURBOC_H

#if !defined(__TURBOC__)
/*
 *   f a r ポインタ用のマクロ
 */
#undef FP_SEG
#undef FP_OFF
#define FP_SEG(fp)  ((unsigned)((unsigned long)(fp) >> 16))
#define FP_OFF(fp)  ((unsigned)(fp))
#define MK_FP(seg,ofs)  (((unsigned long)(unsigned)(seg)<<16) | (unsigned)(ofs))
#endif

#endif  /* __TURBOC_H */
```

リスト８ (xcat.c)

```
/*
 *  xcat.c: 縮小印字モジュールを使ったcat
 *      written by Motohisa Ohno
 */
#include <stdio.h>
#include <stdlib.h>
#include "lprint.h"

FILE *efopen(const char *file, const char *mode)
{
    FILE *fp;

    fp = fopen(file, mode);
    if (fp == NULL) {
        fprintf(stderr, "can't open %s(%s)¥n", file, mode);
        exit(1);
    }
    return fp;
}

void main(int argc, char **argv)
{
    char buf[BUFSIZ];
    FILE *fp;

    if (argc == 1) {
        while (fgets(buf, sizeof buf, stdin))
            xputs(buf);
        return;
    }
    while (--argc) {
        fp = efopen(*++argv, "r");
        while (fgets(buf, sizeof buf, fp))
            xputs(buf);
        fclose(fp);
    }
}
```


ワンラインNEWS
住友金属工業は米リードライト社と提携，薄膜磁気ヘッド事業に参入する．薄膜磁気ヘッドは次世代型HDD用記録再生素子で需要の拡大が見込まれている．

ハードコピー

プリンタを使いたい大きな理由のひとつに，画面の内容を記録しておきたい，ということがあります．ハードコピーモジュールは，多くのアプリケーションプログラマが必要とするプログラムでしょう．

リスト9に，ノーマルモード／モノクロに対応したハードコピーモジュールを示します．

hcopy.cで定義されている関数の仕様は表5のとおりです．hcopy.cを使うときは，lputc.cとともにコンパイル・リンクしなければなりません．

先頭で定義されているVRAMアドレスは，部分コピーを行うために基準値が変更されます．これらを80個の要素(WORDあるいはBYTE)をもつ配列へのポインタという少し複雑な型で定義しているのは，行と桁を別々に指定しやすくするためです．

リスト9 (hcopy.c)

```
/*
 *  hcopy.c: ハードコピーモジュール (ノーマルモード モノクロのみ)
 *          written by Motohisa Ohno
 */
#include <dos.h>
#include <string.h>
#include <conio.h>
#include "turboc.h"
#include "lprint.h"

/* 各種 VRAM アドレス */
static WORD (far *tvram)[80] = (WORD (far *)[80])MK_FP(0xA000, 0);
static WORD (far *avram)[80] = (WORD (far *)[80])MK_FP(0xA200, 0);
static BYTE (far *bvram)[80] = (BYTE (far *)[80])MK_FP(0xA800, 0);
static BYTE (far *rvram)[80] = (BYTE (far *)[80])MK_FP(0xB000, 0);
static BYTE (far *gvram)[80] = (BYTE (far *)[80])MK_FP(0xB800, 0);

#if !defined(__TURBOC__)
/*
 *  MS-C 用のフォントを取得するための関数 (Turbo C++ のサブセット)
 */
static void pc98getfont(unsigned code, void *buffer)
{
    union REGS regs;

    regs.h.ah = 0x14;
    regs.x.bx = FP_SEG(buffer);
    regs.x.cx = FP_OFF(buffer);
    regs.x.dx = code;
    int86(CRTBIOS, &regs, &regs);
}
#endif

/*
 *  テキスト画面全体を正しい VRAM 内容に書換える
 */
static void reform(void)
{
    static WORD (far *tseg)[80] = (WORD (far *)[80])MK_FP(0xA000, 0);
    int col, row;
    WORD code;

    for (row = 0; row < ROWMAX; row++) {
        code = 0;
        for (col = 0; col < COLMAX; col++) {
            if (code) {
                if ((BYTE)code == 0x56 || (BYTE)code == 0x57) {
                    tseg[row][col] &= 0x7F7F;
                    if (tseg[row][col] != code) goto skip;
                }
                tseg[row][col] = code | 0x80;
                code = 0;
            } else {
    skip:;
                code = tseg[row][col];
                if (code >> 8) {
                    tseg[row][col] &= 0x7F7F;
                }
                if ((code >> 8) == 0
                        || (0x09 <= (BYTE)code && (BYTE)code <= 0x0B))
                    code = 0;
            }
        }
    }
}

/*
 *  テキスト画面の指定した位置 (行、桁) の文字フォントを取得する
 */
static void charfont(int row, int col, BYTE *bufp)
{
    BYTE buf[34];
    WORD code, attr;
    int i, j;

    if (row >= ROWMAX || col >= COLMAX) {
        memset(bufp, 0, 16);
        return;
    }
    code = tvram[row][col] & 0x7F7F;
    attr = avram[row][col];
    if ((code >> 8) == 0)
        code += 0x8000;
    else
        code = ((code >> 8) | (code << 8)) + 0x2000;
    pc98getfont(code, buf);
    j = (tvram[row][col] & 0x8080) == 0 ? 2 : 3;
    for (i = 0; i < 16; i++, j += buf[1])
        bufp[i] = buf[j];
    if ((attr & 1) == 0) memset(bufp, 0, 16);
    if (attr & 0x04)
        for (i = 0; i < 16; i++) bufp[i] = ~bufp[i];
    if (attr & 0x08) bufp[15] = 0xFF;
    if (attr & 0x10)
        for (i = 0; i < 16; i++) bufp[i] |= 0x08;
}

/*
 *  テキスト画面の指定した位置 (内部行、桁) の文字フォントを取得する
 */
static void imgtext(int curlin, int curcol, BYTE *bufp)
{
    BYTE fbuf[16], fbuf2[16];
    int i, j;

    charfont(curlin*3/4, curcol, fbuf);
    switch (curlin%4) {
    case 0:
        for (i = 0; i < 12; i++) bufp[i] |= fbuf[i];
        break;
    case 1:
        charfont(curlin*3/4+1, curcol, fbuf2);
        for (i = 0; i < 4; i++) bufp[i] |= fbuf[12+i];
        for (i = 0; i < 8; i++) bufp[4+i] |= fbuf2[i];
        break;
    case 2:
        charfont(curlin*3/4+1, curcol, fbuf2);
        for (i = 0; i < 8; i++) bufp[i] |= fbuf[8+i];
        for (i = 0; i < 4; i++) bufp[8+i] |= fbuf2[i];
        break;
    case 3:
        for (i = 0; i < 12; i++) bufp[i] |= fbuf[4+i];
    }
}

/*
 *  グラフィックス画面の指定した位置 (内部行、桁) のビットイメージを取得する
 */
static void imggraph(int curlin, int curcol, BYTE *bufp, BYTE (far *vaddr)[80])
{
    int i;

    for (i = 0; i < 12; i++)
        bufp[i] |= vaddr[curlin*12+i][curcol];
}

/*
 *  ビットイメージ (8×12) をプリンタイメージ (24×8) に変換する
 */
static void imgconv(BYTE *srcp, BYTE *dstp)
{
    int n, i, j;
    BYTE pat, mask;

    for (n = 0, mask = 0x80; n < 8; n++, mask >>= 1) {
        for (i = 0; i < 3; i++) {
            for (j = 0, pat = 0; j < 4; j++) {
                pat >>= 2;
                if (srcp[i*4+j] & mask) pat |= 0xC0;
            }
            dstp[n*3+i] = pat;
        }
    }
}

/*
 *  ハードコピーモード
 */
static int copymode = HCOPY_PR | HCOPY_TEXT | HCOPY_GRAPH;

```

文字フォントを取得する関数pc98getfontは，縮小印字のために用いたものと同じです．staticとして宣言されているのは，縮小印字モジュールと併用した場合に，シンボルの重複が生じないようにするためですが，実際には，共通の関数として1つにまとめるべきでしょう．

関数reformは，テキストVRAMの内容を正しい内容に設定するものです．VM以降のPC-9801では漢字を表示する際に，左側のVRAMに漢字コードを書き込めば，右側の文字コードに関係なく左側に書き込んだ漢字が表示されます．VMより前の機種では，右側のコードがまちがっていると，あきらかに表示がおかしくなります．このハードコピールーチンでは，テキストVRAMの漢字コードを取得する際に，VRAMの内容にしたがって漢字の左右どちらの桁であるかを判定しているため，テキストVRAMの内容が

表5

hcopy

機能	画面全体のハードコピーをとる．
形式	oid hcopy (int mode)；
説明	テキスト画面とグラフィックス画面を混在させて，画面のハードコピーをとります．ただし，テキスト画面は25行モード，グラフィックス画面は400ラインモードに限ります．テキスト画面やグラフィックスのプレーン（青，赤，緑）について，どれをコピーの対象にするかは引数modeで個別に指定できます．引数modeには表6で示されたシンボル，あるいは複数のシンボルの論理和をとったものを指定できます．ここで対象プレーンとプリンタの種別は別々に切り換えられます．

hcopypart

機能	画面の一部だけのハードコピーをとる．
形式	void hcopypart (int mode, int left, int top, int right, int bottom)；
説明	hcopyと同様の関数ですが，ハードコピーをとる範囲を指定できるようになっています．範囲は，テキスト画面の文字単位で指定し，水平方向は0～79，垂直方向は0～24の値となります．

表6 ハードコピーのモード

シンボル	値	意味
HCOPY_COPY	0000H	最後に指定されたモードでハードコピーをとる．
HCOPY_BLUE	0001H	青プレーンをコピーの対象とする．
HCOPY_RED	0002H	赤プレーンをコピーの対象とする．
HCOPY_GREEN	0004H	緑プレーンをコピーの対象とする．
HCOPY_GRAPH	0007H	グラフィックスをコピーの対象とする．
HCOPY_TEXT	0010H	テキストをコピーの対象とする．
HCOPY_ALL	0017H	テキストとグラフィックスをコピーの対象とする．
HCOPY_PR	0100H	PC-PRプリンタを使う．
HCOPY_NM	0200H	NMプリンタを使う．
HCOPY_SETMODE	8000H	モード指定のみ行い，コピーはしない．

```
/*
 *  ビットイメージをプリンタへ出力
 */
static void loutimage(BYTE *bufp)
{
    int i, j;

    for (i = 0; i < 24; i++)
        if (bufp[i]) break;
    if (i == 24) {  /* empty image */
        _lputs("¥033¥010¥033¥010");
    } else {
        _lputs("¥033" "J0016");
        for (i = 0; i < 8; i++) {
            for (j = 0; j < 3; j++)
                _lputc(bufp[i*3+j]);
            for (j = 0; j < 3; j++)
                _lputc(bufp[i*3+j]);
        }
    }
}

/*
 *  部分ハードコピー
 */
void hcopypart(int mode, int left, int top, int right, int bottom)
{
    int curlin, curcol;
    int i;
    int width, height;
    BYTE buf[24], img[12];

    /* ハードコピーモードの設定 */
    if (mode & ~HCOPY_SETMODE) {
        if (mode & (HCOPY_PR | HCOPY_NM)) {
            copymode &= ~(HCOPY_PR | HCOPY_NM);
            copymode |= mode & (HCOPY_PR | HCOPY_NM);
        }
        if (mode & (HCOPY_ALL)) {
            copymode &= ~HCOPY_ALL;
            copymode |= mode & HCOPY_ALL;
        }
    }
    if (mode & HCOPY_SETMODE) return;

    if (left < 0 || left >= COLMAX
        || top < 0 || top >= ROWMAX
        || right < 0 || right >= COLMAX
        || bottom < 0 || bottom >= ROWMAX
        || left > right || top > bottom) return;

    /* ＶＲＡＭの原点アドレスの設定 */
    tvram = (WORD (far *)[80])MK_FP(0xA000, (top*COLMAX+left)*2);
    avram = (WORD (far *)[80])MK_FP(0xA200, (top*COLMAX+left)*2);
    bvram = (BYTE (far *)[80])MK_FP(0xA800, top*COLMAX*16+left);
    rvram = (BYTE (far *)[80])MK_FP(0xB000, top*COLMAX*16+left);
    gvram = (BYTE (far *)[80])MK_FP(0xB800, top*COLMAX*16+left);

    width = right - left + 1;
    height = (bottom - top + 1) * 4 / 3;

    reform();

    /* PC-PR プリンタと NM プリンタでは改行幅の指定が異なる */
    if (copymode & HCOPY_PR)
        _lputs("¥033T18¥033K");
    else
        _lputs("¥033T16¥033K");

    for (curlin = 0; curlin <= height; curlin++) {
        /* [ESC] が押されたらコピーを中止する */
        if (kbhit() && getch() == 27) break;
        /* 最終行がきりがよければ終了する */
        if (curlin == height && (bottom - top + 1) * 4 == height * 3) break;
        for (curcol = 0; curcol < width; curcol++) {
            memset(img, 0, 12);
            if (copymode & HCOPY_BLUE) imggraph(curlin, curcol, img, bvram);
            if (copymode & HCOPY_RED) imggraph(curlin, curcol, img, rvram);
            if (copymode & HCOPY_GREEN) imggraph(curlin, curcol, img, gvram);
            if (copymode & HCOPY_TEXT) imgtext(curlin, curcol, img);
            if (curlin == height) {
                for (i = (bottom-top+1)*16 - height*12; i < 12; i++)
                    img[i] = 0;
            }
            imgconv(img, buf);
            loutimage(buf);
        }
        _lputc('¥r');
        _lputc('¥n');
    }
    _lputs("¥033A¥033H");
}

/*
 *  全画面のハードコピー
 */
void hcopy(int mode)
{
    hcopypart(mode, 0, 0, COLMAX-1, ROWMAX-1);
}
```


ワンラインNEWS
岩波書店は外国人が日本語を学ぶためのCD-ROMソフトを開発する．ソニーが発売する「DD-1（通称データディスクマン）」を輸出するのに歩調を合わせた．

まちがっていると，正しいハードコピーが得られません．reformでは，漢字の左の桁はMSBを０にし，右の桁はMSBを１にすることで,左右の判定を正しく行えるようにしています．

なお，外字を全角として表示する場合は，必ず左右に文字コードが書き込まれていなければなりません．古いPC-286／386シリーズでは，外字の左半分だけを書き込めば，外字全体が表示されますが，PC-9801シリーズでは片方だけを書き込むと，半角だけが表示され，若干の制限はあるものの半角外字として利用することができます．これを利用したプログラムもあるため，このハードコピーモジュールではPC-9801シリーズの仕様にあわせてあります．

charfont関数は，pc98getfontを使って指定された位置の文字フォントを取得する関数です．この文字フォントは，指定された位置の属性によっても変更されます．ただし，黒で表示された文字は消去されません．これを行うためには，グラフィックス画面上に黒のテキストで書かれた文字について，黒が優先されてそのパターンを抜く処理をしなければなりません．その必要が生じるアプリケーションはまれでしょうし，そのための手間と実行時のオーバーヘッドを考慮すると，必ずしも必要な処理ではありません．

imgtextとimggraphは，指定された内部行，桁のビットイメージを指定されたバッファと論理和をとっていくものです．imgconvは，縮小印字のときと同様に，内部で横方向に並べられているビット列を縦方向に変換して，ビットイメージの印字形式にするものです．

loutimageは，画面の横８ドット（プリンタ上では16ドット）分をプリンタに出力する関数です．ここでは，出力するイメージが全部０であるかどうかを調べて，全部０であれば，８ドットスペースを２回出力して印字時間を短縮しています．このため，ハードコピー中は，ドットスペースが有効になる漢字モードにしてあります．

hcopypartが，ハードコピーのメインの関数でハードコピーのモードを設定した後，ハードコピー処理を行います．範囲の指定は，指定された行と桁に応じてテキストVRAMとグラフィックスVRAMの原点アドレスを変更し，そこからの相対値によってコピーデータを取得するようにしています．印字処理は，内部行によってカウントされます．これはプリンタのビットイメージが24ドットなので，一度に12ラインごとのデータしか出力できず，テキストの16ライン（ドット）を基準にできないためです．実際に処理されているテキスト行は，内部行×３／４となります．

hcopyは，画面全体のハードコピーをとる関数ですが，hcopypartの範囲として全画面の領域を指定しているだけです．

ハードコピーのテストプログラムをリスト10に示します．このプログラムは，Turbo C++のBGIを使用していますのでMS-Cでは使えません．hctest.cは，hcopy.cとlputc.cおよびgraphics.libとともにコンパイル・リンクし，実行する際はPC98.BGIをカレントディレクトリにコピーしておいてください．MS-Cでは，別のテキスト関数やグラフィックス関数を使って画面を設定してからhcopyを呼び出してください．

リスト10 (hctest.c)

```
/*
 * ハードコピーのテスト(Turbo C++のみ)
 */
#include <graphics.h>
#include <stdio.h>
#include <conio.h>
#include "lprint.h"

void main()
{
    int gd, gm;
    int i;

    gd = DETECT;
    initgraph(&gd, &gm, "");
    for (i = 0; i < 640; i += 8) {
        setcolor(i%7 + 1);
        line(i, 0, 639-i, 399);
    }
    for (i = 0; i < 400; i += 8) {
        setcolor(i%7 + 1);
        line(639, i, 0, 399-i);
    }
    for (i = 0; i < 100; i++) {
        textcolor(i%7 + 1);
        cprintf("色番号%d, ", i);
    }
    hcopy(HCOPY_ALL);
    closegraph();
}
```

98活用アセンブラ技法

タイマ編
第1回

タイマBIOSの補強

星野 操

タイマBIOSは，複数のプロセスで共用できない，繰り返し処理が正確にできない，繰り返し処理をする場合に割り込みルーチンでタイマの再設定が必要になるなど，PC-9801のBIOSの中でも問題点が多い．しかし，同じPC-9801でもハイレゾモードのタイマBIOSは，かなり機能を強化したものになっている．そこで，今回はこのハイレゾモードのタイマBIOSの機能を，ノーマルモードのPC-9801に付加するための常駐プログラムを作成した．

タイマBIOSの動作

セットして一定時間たつと割り込みが発生

さて，今回はタイマBIOSの補強を行ってみます．

PC-9801の（ノーマルモードの）タイマBIOSは，BIOSの中でももっとも評判の悪いものといえます．PC-9801のBIOSでは，INT1CHを呼び出してタイマをセットすると，指定した時間(10ms単位)が経過した後に指定したルーチンが呼び出されます．

もう少し詳しく説明しましょう．タイマの設定は，INT1CH/AH=2の呼び出しで行います．このとき，CXにディレイする時間（10ms単位）を，ES:BXにタイムアップしたときに起動するアドレスを設定しておきます．時間は低位のワークエリアに，アドレスはINT7の割り込みベクタに設定され，タイマ割り込みが許可され，タイマが方形波モード(10msごとに割り込みがかかるモード)に設定されます．

10msたつと割り込みが発生し，BIOSの中に制御が移ります．ここでワークエリアのカウンタがデクリメントされ，0でなければPICにEOIコマンドを発行した後にINT1CH/AH=3を実行し，0であれば(タイムアップすれば)タイマ割り込みを禁止し，EOIをPICに送った後にそのままINT7を実行します．

INT1CH/AH=3で何が実行されるかというと，タイマに10msの時間定数を書き込み，割り込みを許可するだけです．ですから，とくに割り込みを分けるほどの細工をしているわけではありません．タイマは，ワンショットモードで起動しているわけではありませんから，時間定数を書き込まなくても割り込みは継続して発生します．逆に時間定数を書き込むことによって，時間間隔の起点が後ろにずれるので，実際の割り込み間隔は10msより長くなることになります．

タイマBIOSの問題点

複数プロセスの共用，ワンショットモード……

さて，以上のようなBIOSの仕様の問題点をあげると，次のようなものになるでしょう．

複数のプロセスで共用できない

これが，タイマBIOSの最大の問題点といえるでしょう．常駐プログラムなどを使用する場合に，定期的に割り込みルーチンを呼び出す必要があります．ところが，この仕様だと1つのプロセスでしか，タイマ割り込みを使用することはできません．カウンタや割り込みベクタなどは，1セットしか用意されていませんから，複数のイベントを扱えないのは当然です．あとからタイマを使用するコマンドを走らせると，前のタイマ呼び出し要求が無視されてしまいます．

たとえば，プリンタスプーラなどを常駐させていると，ほかの常駐プログラムではタイマを使用することができないのです．状況は表のプログラムでも同様で，通信プログラムなど，タイムアウト処理にタイマを使っている場合には，通信プログラムとスプーラとどちらかの処理が止まってしまうことになります．

このため，PC-9801ではVSYNC割り込みをこの目的で使用することがありますが，これはビデオ関係のBIOSが同期を取るために自前で使用しており，画面スプリットなどの特定のBIOS呼び出しをすると，割り込みが止まってしまうことになります（これによって，FEPを起動したときに割り込みが止まるということがよくあります）．また，割り込みを継続的に発行するために，リセットポートというポートに出力をする必要があり，複数のプロセスでこれを共用する場合には，このポートをたたくタイミングが問題になります．場合によっては，割り込みの再入がおきてしまいます．

以上のように，PC-9801では複数のプロセスからタイマを使用することは，非常にやりにくいようになっているというわけです．

繰り返し処理が正確にできない

仕様は，基本的にワンショットモードとなっています．一定時間たったときに割り込みを発生させるというものです．しかし，その一定時間にしても，10msのハードウェア割り込みを複数回発生させる必要があり，そのたびに前記のように，割り込み応答のディレイが積算されるので，応答時間が正確であるとはいえません．

ところで，実際にタイマを使うことを考えてみると，ワンショット割り込みですむという処理は意外と多くありません．あえてあげるとしたら，CPUの速度に依存しないディレイを取る場合ぐらいでしょうか．そのほかの大部分の場合は，何度も何度も繰り返し処理を必要とするものです．

たとえば，前記のプリンタスプーラなどは，処理にタイマ割り込みを使用しています．一定時間ごとにプリンタポートを見に行き，プリンタの準備ができていれば出力を行うのです．ですから，プリントをはじめてからすべてのプリントが終了するまで，一定間隔ごとに繰り返し割り込みサービスが必

要だということになります．ほかの例でも，たいてい繰り返しサービスをすることになるでしょう．

プリンタスプーラの例では，あまりたいした影響はありませんが，かなり正確に一定間隔ごとのサービスが必要になるという場合もあります．サービスごとの間隔が正確であるというよりは，ならした場合にトータルでの割り込み発生間隔が正確であることが要求されるというものです．たとえば，正確に1秒ごとにサービスを発生させ，そのたびごとに内部カウンタを進めて，経過時間をはかろうというような場合です．

このような場合，サービスとサービスの間の時間が厳密である必要はさほどありません（もちろん，正確であるにこしたことはありません）．実際，割り込みが発生してから受け付けられるまでの時間には，どうしてもばらつきがでるので，それほど正確にはできないのです．それよりも，誤差が累積しないことのほうが大切です．多少間隔がふらついても，平均値が安定していることが必要なのです．

ところが，ワンショットタイプの割り込みの場合だと，この「長期的な間隔一定」は満たされません．割り込みがかかってから，再度割り込み設定のBIOSコールを発行するまでの無駄時間は累積されていってしまうのです．もちろん，この無駄時間は安定しませんから，簡単に補正するというわけにもいきません．加えて，PC-9801のBIOSの場合には，ワンショットのディレイ時間自体も安定しているわけではないのです．

割り込みルーチンでタイマの再設定が必要

前記のように，タイマを使用するたいていのルーチンは，繰り返しサービスを必要とします．ワンショットタイプの割り込みで，繰り返しサービスを行おうとすると，どうしても割り込みルーチンの中で，割り込み再設定のコマンドを発行しないといけません．つまり，割り込みサービスルーチンの中でINT1CHを呼び出す必要があるのです．

これは，あまりにも美しくない仕様といえます．割り込み処理ルーチンは，なるべく内部で閉じているほうがよいに決まっていますから，BIOSコールの発行を要求するというのは好ましいとはいえません．

ここで生じる問題の多くは，割り込み処理が基本的にワンショットである，ということに起因します．ワークエリアなどの問題に加えて，時間処理の観点からもワンショットモードの場合は，複数のタイマ要求を管理できないのは明らかでしょう．

もちろん，ワンショットタイプのサービスの利点も存在しないわけではありません．とくに，多重割り込みが発生しないというのは大きな利点です．タイムアウトが発生した時点でタイマ割り込みは禁止されますから，再度割り込み設定を行わない限り，割り込みは発生しません．

それに対して，定期的に割り込みが発生するようなサービスを用意した場合には，割り込みの処理が終了しないうちに，またタイマのオーバーフローが発生すると，再度割り込みがかかることになります．割り込みコントローラの設定がすんでいなければ，割り込みを1つ抜かすことになりますし，そうでないと割り込みの再入がおきることになります．同様な問題は，キーボードの場合にも存在し，[STOP][COPY]キーのハンドラは，割り込みルーチンの処理時間が長い場合には，再入についても配慮しなければいけません．ですから，もしこのような処理を期待するならば，利用するプログラムの側でも十分に再入配慮をする必要があります．

以上のように，いろいろな問題があります．このため，プログラムのほうではBIOSを使用せず，自分でタイマの設定とハードウェア割り込みルーチンをもち，ハードウェア割り込み自体をトラップしている場合も少なくありません．

IBM PCのタイマBIOS

タイマ割り込みは周期的に発生する

比較的使いやすいといわれる，IBM PCのタイマ関係の仕様についても触れておきましょう．こちらは，とくに割り込み関係のサービスルーチンというものはもっていません．タイマ割り込みは，周期的に発生するように設定されており，ハードウェア割り込みが発生すると，サービスルーチンが別のソフトウェア割り込みを呼び出すようになっています．

昔のIBM PCは，リアルタイムICのような石をもっていなかったので，時間管理はこのタイマ割り込みを用いていました．最初にデフォルトで設定されるソフトウェア割り込みルーチンの中で，ワークエリアのカウンタをカウントアップするわけです．ハードウェア割り込みの再設定は，ソフトウェア割り込みを呼び出したあとの割り込みサービスルーチンで行っていると考えてよいでしょう．

ユーザープログラムが定期的な割り込みを使用する場合には，このソフトウェア割り込みをトラップすることになります．自前の処理をすませた後，

ワンラインNEWS
ハル研究所は2台のパソコン間で双方向データ転送のできるソフト「転送くん」を発売．ノート型パソコンの普及で，複数のパソコンをもつ個人をターゲットにしている．

リスト1

```
;-----------------------------
; timer: timer interrupt handler
;
; $Log: RCS/timer.asm $
; revision 1.1 TAN 91/09/05 07:31:22
; Initial revision
;
;
;-----------------------------

        dosseg
        .model  small,pascal

        include condx.inc
        include utils.inc

        masm51

        .stack  2000h

; for interrupt
        .code
        public  my_id,interrupt_list,init_func

my_id   dw      2584h

int_work        struc
int_num         dw      ?
int_func        dw      ?
int_old         dd      ?
int_work        ends

interrupt_list  label   byte
i08     int_work        <8,int_08,0>
i1c     int_work        <1ch,int_1c,0>
        dw      0

        .data
divide_value    =       8
timer_repeat    =       1

timer_work      struc
forward_link    dd      ?               ; forward link
timer_value     dw      ?               ; decrement count

timer_count     dw      ?               ; timer work initial value
timer_routine   dd      ?               ; execute routine on timeout

backward_link   dd      ?               ; backword link pointer
timer_status    db      ?               ; overflow flag, repeat mode
timer_divide    db      ?               ; timer divide work
timer_work      ends

timer_tail      label   timer_work
        dd      0
        dw      0,0
        dd      0
        dw      timer_base
        dw      _DATA

timer_base      label   timer_work
        dw      timer_tail
        dw      _DATA

in_interrupt    dw      0               ; interrupt execution flag
require_count   dw      0               ; service require count

single_count    dw      0               ; single event interrupt count
single_work     dw      0               ; divide count

        .code

int_08:
        ; timer interrupt handler
        cli                             ; for insureance

        ; set ds value
        push    ds
        push    ax
        mov     ax,_DATA
        mov     ds,ax

        ; increment service count
        inc     require_count

        ; send eoi first
        mov     al,20h
        out     0,al

        ; check in service
        cmp     in_interrupt,0
        @if     nz
                pop     ax
                pop     ds
                iret                    ; short cut
        @endif

        ; save other registers
        push    es
        push    bx

int_08_loop:
        ; main service loop
        sti

        ; check single event
        cmp     single_count,0
        @if     nz
                dec     single_work
                @if     z
                        mov     single_work,divide_value
                        dec     single_count
                        @if     z
                                ; call single event
                                int     7
                        @endif
                @endif
        @endif

        push    ds                      ; load pointer value
        pop     es
        mov     bx,offset timer_base
        @begin
                ; advance pointer
                les     bx,es:[bx].forward_link

                ; check end of list
                mov     ax,word ptr es:[bx].forward_link
                or      ax,word ptr es:[bx].forward_link+2
        @while  nz
                dec     es:[bx].timer_divide
                @if     nz
                        @continue       ; short cut
                @endif

                mov     es:[bx].timer_divide,divide_value
                dec     es:[bx].timer_value
                @if     nz
                        @continue       ; short cut
                @endif

                ; save registers used
                push    ds
                push    es
                push    bx

                ; emulate interrupt
                pushf
                call    es:[bx].timer_routine

                ; resume register value
                pop     bx
                pop     es
                pop     ds

                ; check repeat mode
                @mc
                        test    es:[bx].timer_status,timer_repeat
                        @cond   nz
                        or      ah,ah   ; check return value
                        @cond   nz
                @if
                        ; repeat again
                        mov     ax,es:[bx].timer_count
                        mov     es:[bx].timer_value,ax
                @else
                        ; unlink pointer
                        push    si
                        push    ds
                        push    bx
                        push    es

                        lds     si,es:[bx].backward_link
                        les     bx,es:[bx].forward_link

                        mov     word ptr es:[bx].backward_link,si
                        mov     word ptr es:[bx].backward_link+2,ds

                        mov     word ptr ds:[si].forward_link,bx
                        mov     word ptr ds:[si].forward_link+2,es

                        pop     es
                        pop     bx

                        ; clear forward link, show inactive work
                        sub     ax,ax
                        mov     word ptr es:[bx].forward_link,ax
                        mov     word ptr es:[bx].forward_link+2,ax

                        pop     ds
                        pop     si
                @endif
        @repeat

        ; check service count
        cli
        dec     require_count
        @if     nz
                jmp     int_08_loop     ; repeat onece more
        @endif

        mov     in_interrupt,0          ; reset flag
        pop     bx
        pop     es
        pop     ax
        pop     ds

        iret

int_1c:
        ; timer bios service routine
        @mc
                cmp     ah,2
                @cond   b
        @or
                cmp     ah,6
                @cond   a
        @if
                ; not my service
                jmp     cs:i1c.int_old
```

もとの割り込みルーチンにジャンプするか，あるいはもとのルーチンを呼び出してから自前の処理をします(スタックの関係から，前者の処理のほうが望ましいことは望ましい).

ハードウェア関係の処理は，デフォルトでサポートされているので，ユーザープログラムでは何も考えずに割り込みのチェインを実行すればよいのです．もちろん，ここで実行されるのは継続的な割り込み処理であり，ディレイを求めるワンショット的な使い方は困難になります．

PC-9801のVSYNC割り込みも，これに比較的似ています．しかし，割り込み再設定を行う部分が，デフォルトで用意されていないという点が大きく違います．このため，割り込みを利用するすべてのルーチンで再設定の処理を行う必要があり，複数のルーチンで共用する場合には，この点で競合が発生する可能性があります．

IBM PCの仕様で，一度別のソフトウェア割り込みを実行し，そちらのほうを公開しているというのは，非常に優れているといえます．高速で処理が終了するルーチンは，このチェインを実行するだけで十分です．これに対して，時間がかかる処理を必要とする場合もあります．ホットキーを検出し，それでプログラムをポップアップするような場合，チェックする契機をタイマ割り込みに求めることがあります．この場合，ポップアッププログラムを終了するまで，いわば割り込み処理ルーチンを継続することになるわけですから，単なるトラップでは不都合です．

このような場合に，はじめてハードウェア割り込みルーチンをトラップすることになります．そのほかのタイマ割り込みルーチンの処理をすませてから，再入処理をもった処理ルーチンを実行することで，割り込み動作を継続したまま，結果的に長時間の割り込み処理が可能となるのです．ただし，この場合でも複数のルーチンが，このような処理を共用することは困難です．

常駐プログラムなど書いてみるとすぐにわかることですが，IBM PCのようないわば簡単な仕様であっても，たいていの場合には目的を十分達成できるものです．

PC-9801ハイレゾモードのタイマBIOS

ノーマルモードを大幅に強化

もちろん，NECのほうもこれらの問題点は十分把握していたとみえて，「ハイレゾモード」のBIOSではかなりの強化がなされています．ハイレゾモードのBIOSでは，このタイマサポートのほかに，キーボード処理などに関してもずいぶんと改良がされています．

しかし，ご存知のようにハイレゾモードは普及しているとはいえません．このように大きなギャップを作ってしまったことが災いしているのだと思われます．BIOSのみならず，ハードウェア的な仕様(割り込みの使い方やメモリマップに至るまで)までも「同時に」変更するのは大きな問題です．

Windowsなどを使うことを考えると，現在の640×400では解像度がたりないのは明らかです．とくに，全角文字を使ったメニューを見ていると情けなくなります．ハイレゾモードの普及が期待されますが，そのためにはBIOSやハードウェアの構成はそのままに，ビデオ関係だけ解像度を上げるようなマシンを出さないといけないのではないでしょうか．現在のハイレゾマシンは，ハイレゾとノーマルのモードの「デュアルマシン」に過ぎませんから．

マルチイベントの処理

という話はさておいて，いずれにしろハイレゾモードのタイマBIOSは比較的よくできています．この仕様を少し調べておきましょう．ハイレゾモードでは上記シングルイベントの割り込みをサポートするほかに，マルチイベントの処理も可能となっています．この場合，イベントの数だけワークエリアが必要なのは明らかですが，このワークエリアは，ユーザープログラムのほうで用意し，BIOS呼び出し時にアドレスを渡すようにします．

マルチイベントのサポートは，ワンショットモードのほかに，リピートモードのサポートがあり，また登録されている割り込みを停止することも可能です．リピートモードでは返り値(AHレジスタ)の値によって，割り込みを継続するか否かを決定することができます．

タイマは，10msごとに継続して割り込みがかかるようになっています．シングルイベントの割り込みは，このハードウェア割り込みによってカウンタをデクリメントします．タイムアップしたときにINT7の割り込みを発生させます．マルチイベント用のワークエリアはリスト構造になっています．割り込みによってワークエリアのカウンタをデクリメントし，タイムアップによって割り込みを疑似的に発生(PUSHFとFAR CALLでエミュレート)させます．処理が終わったら，リピートモードで継続される返り値でなければリストからワークエリアを切り離します．

マルチイベントでは，複数のワークエリアの処理をする必要があるので，実際に疑似割り込みを発生させるまで

ワンラインNEWS
エルゴソフトは東芝 J-3100シリーズ用の表計算ソフト「EG Calc」を発売した．ネットワーク環境で複数のユーザーがファイルを共有できるのが特徴

のディレイが生じます．この影響を少なくするために，カウンタの値が小さいものほど，リストの先頭にくるようにしてあります．しかし，いずれにしろ，長期的な割り込み間隔の平均値は安定しているといえます．

シングルイベントの処理

これに対して，シングルイベントのほうには多少の問題があります．ノーマルモードの場合には，BIOSの呼び出しがあってからタイマの初期化が行われるので，その意味での時間の不安定さはありません．しかし，ハイレゾモードだとタイマはシングルイベントの呼び出しとは非同期に10ms間隔で走っているので，要求したディレイに対して0～10msの間での時間的な不安定さが発生します．

しかし，いずれにしろ便利であることはたしかです．DOSのPRINTコマンドでも，ハイレゾモードであるかどうかを判定し，ハイレゾであればマルチイベントを使用するようになっています．ただ，ノーマルモードではBIOSは使用されず，直接ハードウェア割り込みを使用しています．

タイマBIOS補強プログラムTIMER.ASM

ノーマルモードでハイレゾモードの仕様を実現

ノーマルモードでも，ハイレゾモードと同様な仕様を常駐プログラムの形で実現することができます．今回は，これをインプリメントしてみましょう．この場合，シングルイベントもサポートしておけば，これを使用しているプログラムとの共用も可能ですが，残念ながらノーマルモードでタイマを使用する多くのプログラムは，ハードウェアを直接触っているので共存でき

```
		@endif

		; save registers
		sti
		push	ds
		push	es
		push	bx
		push	cx

		; set data register
		push	ax
		mov	ax,_DATA
		mov	ds,ax
		pop	ax

		cmp	ah,2
		@if	z
			; single event timer set routine
			cli

			mov	single_work,divide_value
			mov	single_count,cx

			; routine set
			push	ds
			sub	cx,cx
			mov	ds,cx
			mov	ds:[4*7],bx
			mov	ds:[4*7+2],es
			pop	ds

			sti
		@else
		cmp	ah,3
		@e_if	z
			; timer cancel
			cli

			; check forward link work
			mov	ax,word ptr es:[bx].forward_link
			or	ax,word ptr es:[bx].forward_link+2
			@if	nz
				; active work, unlink
				push	es
				push	bx
				push	ds
				push	si
				lds	si,es:[bx].backward_link
				les	bx,es:[bx].forward_link

				mov	word ptr es:[bx].backward_link,si
				mov	word ptr es:[bx].backward_link+2,ds

				mov	word ptr ds:[si].forward_link,bx
				mov	word ptr ds:[si].forward_link+2,es

				pop	si
				pop	ds
				pop	bx
				pop	es

				sub	ax,ax
				mov	word ptr es:[bx].forward_link,ax
				mov	word ptr es:[bx].forward_link+2,ax
			@endif

			sti
		@else
		cmp	ah,5
		@e_if	be
			; timer regist (one shot/repeat mode)
			cli

			; set work value
			cmp	ah,4
			@if	z
				mov	al,0
			@else
				mov	al,timer_repeat
			@endif
			mov	es:[bx].timer_status,al
			mov	es:[bx].timer_divide,divide_value
			mov	ax,es:[bx].timer_count
			mov	es:[bx].timer_value,ax


			; link
			mov	ax,word ptr timer_base
			mov	word ptr es:[bx].forward_link,ax
			mov	ax,word ptr timer_base+2
			mov	word ptr es:[bx].forward_link+2,ax
			mov	word ptr es:[bx].backward_link,offset timer base
			mov	word ptr es:[bx].backward_link+2,ds

			mov	word ptr timer_base,bx
			mov	word ptr timer_base+2,es

			push	es
			push	bx
			les	bx,es:[bx].forward_link
			pop	word ptr es:[bx].backward_link
			pop	word ptr es:[bx].backward_link+2

			sti
		@endif

		pop	cx
		pop	bx
		pop	es
		pop	ds

		iret

init_func	proc	far
		; invoke timer interrupt
		mov	dx,19968/divide_value	; 8MHZ
		in	al,42h
		test	al,20h
		@if	z
			mov	dx,24576/divide_value	; 10MHZ
		@endif

		cli
		mov	al,36h
		out	77h,al
		jmp	$+2
		jmp	$+2
		mov	al,dl
		out	71h,al
		jmp	$+2
		jmp	$+2
		mov	al,dh
		out	71h,al

		; mask clear
		in	al,2
		and	al,not 1
		jmp	$+2
		jmp	$+2
		out	2,al

		ret
init_func	endp

		end
```

ません．ですから，過大な期待をすることはできませんが，少なくとも自分でタイマを使うプログラムを作る場合には有効でしょう．

タイマ割り込みをサポートする部分をTIMER. ASMに示します．これと常駐をサポートするRES2. ASMを，アセンブル・リンクしてプログラムを作成します．常駐プログラムのスケルトンは，以前にRESIDENT. ASMを掲載したことがありますが，これは不確定のユーザーインターフェースを1つもつというタイプのものでした．

INT29Hをトラップして，バックカラーをサポートする8回目の連載のときにもこれを使用しましたが，このスケルトンの仕様は，決まった割り込みを(複数)サポートするような常駐プログラムには不向きといえます．ですから，今回多少の変更を加えました．

PUBLIC宣言

プログラムの最初にPUBLIC宣言があります．my_id, interrupt_list, init_funcの3つが宣言されています．最初のmy_idは，常駐idを示すもので，INT2FHの割り込みをデコードするために使用されます．RESIDENT. ASMでは，これを定数として記述していたのですが，それだと処理部分を書き換えるたびに，RESIDENTのほうも書き換えないといけないので不都合でした．今回，これを変数として取ることでスケルトン部分の変更を不要にしています．

interrupt_listは，常駐プログラムがサポートする割り込み番号と処理ルーチンのアドレス，そして元のルーチンのアドレスを格納する配列の先頭アドレスです．実際の構造体の内容は，int_workによって示されています．同様なものは，RES2にも記述されています．

この構造体を利用して，ここではINT8とINT1CHの2つの割り込みルーチンをトラップすることにします．もちろん，INT2FHはRES2のほうでデフォルトでトラップされます．リストの先頭は割り込み番号0で示すことにします．各構造体にラベルをつけているのは，割り込みチェインをする場合に，元の割り込みアドレスをアクセスできるようにするためです．なお，これは割り込みルーチンで処理しやすいように，コードセグメントにおきます．

リスト2

```
;----------------------------
; res2: resident handling part
;
; $Log: RCS/res2.asm $
; revision 1.1 TAN 91/09/05 07:31:30
; Initial revision
;
;
;----------------------------

        dosseg
        .model  small,pascal

        masm51                          ; delete if masm

        include condx.inc

        include utils.inc
        include dosfile.inc

        .code
        extrn   my_id:word
        extrn   interrupt_list:byte
        extrn   init_func:far

int_work        struc
int_num         dw      ?
int_func        dw      ?
int_old         dd      ?
int_work        ends

;----------------------------
; int2f: mulitiplexed interrupt entry
; argument:
;       AX: ID
;       BX: function code
;          0 - check residence
;          1 - release program
; return:
;       DX: psp of program (function 0/1)
;       AX: broken

        .code
vect2f  dd      ?

_int2f:
        ; first, check interface ID
        cmp     ax,cs:my_id
        @if     nz
                jmp     vect2f          ; chain to old interrupt
        @endif

        sti                             ; enable interrupt
        push    ds
        mov     ax,_DATA                ; set data segment value
        mov     ds,ax

        ; branch with function code
        or      bx,bx
        @if     z
                ; check residence, return psp
                mov     dx,$psp         ; $psp is a system variable
        @else
        dec     bx
        @e_if   z
                ; release program
                push    ds

                ; resume all vector
                mov     bx,offset interrupt_list
                @begin
                        mov     ax,cs:[bx].int_num
                        lds     dx,cs:[bx].int_old
                        add     bx,size int_work

                        or      ax,ax
                @while  nz
                        dos     <25h,,>
                @repeat

                ; release int2f vector
                lds     dx,vect2f
                mov     al,2fh
                dos     <25h,,>

                pop     ds

                ; then free environment
                mov     es,$psp         ; first, get environ seg
                mov     ax,es:[2ch]
                mov     bx,ax
                dec     ax
                mov     es,ax
                mov     ax,es:[1]       ; check owner of environ seg
                cmp     ax,$psp
                @if     z
                        ; environment is not released
                        mov     es,bx
                        dos     <49h,,>         ; free environ
                @endif

                ; return psp to release me
                mov     dx,$psp         ; $psp is a system variable
        @endif

        pop     ds                      ; resume ds value
        iret

;----------------------------
; initialize routines
; define intialize code segment
;

        .data
```


ワンラインNEWS
日本錬水は次々世代の半導体である64MビットDRAMの洗浄に利用できる超純水の製造技術を開発．試験用プラントも完成させたことから販売活動を本格化させる．

```
end_pos db      0                       ; show the end of data segment
        public  end_pos

r_option        db      '-r'            ; option string definition
dup_message     db      'program is already installed',0dh,0ah,0
not_res_message db      'program not installed',0dh,0ah,0
rel_ok_message  db      'program correctly released',0dh,0ah,0
correct_message db      'program correctly installed',0dh,0ah,0
arg_err_message db      'argument range error',0dh,0ah,0
vect23  dd      ?                       ; for ctrl_c vector

init_seg        segment 'stack'         ; special segment for initialize
        assume  cs:init_seg             ; cs: init_seg

;----------------------------
; strncmp: compare two strings
; arguent:
;       p1, p2: buffer address
;       n: number to compare
; return:
;       ax: result, 0 for equall, nz for not equall

func    strncmp <p1, p2, n> <si di>
        mov     dx,0                    ; dx for flag
        mov     si,p1
        mov     di,p2
        push    ds                      ; setup es
        pop     es

        @begin
                cmp     n,0
        @while  nz
                lodsb                   ; compare characters
                scasb
                @if     nz
                        mov     dx,1    ; non zero value
                        @break
                @endif

                or      al,al           ; check string tail
                @if     z
                        @break
                @endif
                dec     n               ; decriment counter
        @repeat

        mov     ax,dx                   ; set return value
        ret
endf

;----------------------------
; teminate: terminate function
; argument:
;       status: exit status
;
func    terminate       <status>
        ; restore int23 vector
        lds     dx,vect23
        mov     al,23h
        dos     <25h,,,>

        ; terminate
        mov     ax,status
        dos     <4ch,,,>
endf

dummy_int:
        iret

;----------------------------
; initial: initialize part
;

initial:
        @setup                          ; parse command line

        ; disable ^c interrupt
        mov     al,23h                  ; get ^c vector
        dos     <35h,,,>
        mov     word ptr vect23,bx      ; save it
        mov     word ptr vect23+2,es

        push    ds                      ; set dummy interrupt entry
        push    cs
        pop     ds
        mov     dx,offset dummy_int
        dos     <25h,,,>
        pop     ds

        push    ds
        mov     cx,_TEXT
        mov     ds,cx
        assume  ds:_TEXT
        mov     ax,my_id                ; then check residence
        pop     ds
        assume  ds:_DATA

        mov     bx,0
        mov     dx,bx
        int     2fh

        ; duplicate check
        or      dx,dx
        @if     nz                      ; already installed
                ; check R option
                mov     bx,_argv
                strncmp <offset r_option,[bx+2],2>      ; compare option
                @if     nz              ; reinstall error
                        mov     dx,offset dup_message
                        @puts <dx>

                        terminate <1>   ; error code
                @endif

                ; release program
                push    ds
                mov     cx,_TEXT
                mov     ds,cx
                assume  ds:_TEXT
                mov     ax,my_id
                pop     ds
                assume  ds:_DATA
                mov     bx,1
                int     2fh

                ; and release program area
                mov     es,dx
                dos     <49h,,,>

                ; message
                mov     dx,offset rel_ok_message
                @puts   <dx>

                ; terminate successfully
                terminate <0>
        @endif

        ; check -r flag
        mov     bx,_argv
        strncmp <offset r_option,[bx+2],2>
        @if     z
                mov     dx,offset not_res_message       ; not yet installed
                @puts   <dx>

                terminate <1>           ; error return
        @endif

        ; install vector
        push    ds
        mov     cx,_TEXT
        mov     ds,cx
        assume  ds:_TEXT

        mov     al,2fh                  ; save 2fh vector
        dos     <35h,,,>
        mov     word ptr vect2f,bx
        mov     word ptr vect2f+2,es

        mov     dx,_TEXT                ; set 2fh vector
        mov     ds,dx
        mov     dx,offset _int2f
        dos     <25h,,,>

        ; and install other interrupts
        mov     si,offset interrupt_list
        @begin
                mov     ax,[si].int_num
                or      ax,ax
        @while  nz
                dos     <35h,,,>
                mov     word ptr [si].int_old,bx
                mov     word ptr [si].int_old+2,es

                mov     dx,[si].int_func
                dos     <25h,,,>

                add     si,size int_work
        @repeat

        pop     ds
        assume  ds:_DATA

        ; message output
        mov     dx,offset correct_message
        @puts <dx>

        ; call sub function
        call    init_func

        ; stay resident
        mov     dx,offset dgroup:end_pos        ; calculate paragraph size
        add     dx,15                   ; data segment length
        mov     cl,4
        shr     dx,cl

        add     dx,_DATA                ; add data segment offset
        sub     dx,$psp

        mov     al,0                    ; success flag
        dos     <31h,,,>                ; keep process

init_seg        ends

        end     initial
```

ワンラインNEWS
オリンパス光学工業はアクセス速度が最速の3.5インチMOディスクドライブ「MOS300E」を開発・販売を開始した。

定数と構造体の定義

次に定数と，タイマ用の構造体の定義があります．まず，divide_valueというのは，ハードウェア割り込み間隔を指定するものです．実際の割り込み処理のリゾリューションは10msとしても，ハードウェア割り込み間隔はこれより短くしておいたほうが，シングルイベントの精度が上がります．また，マルチイベントの場合も同時に割り込みが発生する確率が減るので，応答性が上がる可能性があります．反面，割り込みオーバーヘッドは増加します．ここで8と指定してあるのは，10/8msの間隔で割り込みを発生させるというものです．timer_repeatというのは，マルチイベントの割り込みでワンショットモードかリピートモードかを区別するためのフラグです．

タイマ用の構造体は，ハイレゾのそれを拡張したものです．ハイレゾのものは先頭6バイトをBIOS用のワークエリア，そのあと2バイトをカウント値，それから4バイトで処理ルーチンを示しています．今回は，そのあとにリンク用のワークエリア，タイマ処理関係のワークエリアで6バイトを専有しています．timer_statusはワンショット／リピートの区別をするフラグで，timer_divideは前記のように，ハードウェア割り込み間隔を短くしたために必要となる変数です．

双方向リンクの採用

ハイレゾモードのワークエリアは，一方向の単純なリンク構造になっています．これに対して，今回のプログラムでは双方向リンクを採用しています．双方向リンクは，リンクのメインテナンスの手間がかかります．少なくとも，単方向のリンクの2倍の変数を処理する必要がありますし，そこに設定する「内容」を管理するために，実質3～4倍の手間がかかるといえます．そのかわり，リンクを「外す」というような処理は，はるかに簡単にできます．

INT1CH/AH=3では，登録されている割り込みをリンクから外す処理をします．ここで単方向リンクだと，リンクを全部たどって目的の部分を探す必要があります．一方，双方向リンクだと，前のワークエリアと次のワークエリアの位置をすぐに把握できるので，直接リンクを外すことができます．ただし，そのワークエリアがリンクにつながっていない場合についても，多少配慮しなければいけません．

また，単方向リンクではベースのポインタを1つもち，リンクの最後はリンクポインタが，0 (NULL)であることによって示すのが普通です．リンクを外す場合には，前のリンクポインタに外すワークエリアのリンクポインタの内容を設定すればよいのです．これに対して，双方向リンクを利用する場合には，現在のリンクポインタの内容がNULLであった場合に，特別な処理をしなければいけません．

そのようなめんどうを避けるために，今回はベースのほかにリストの終わりを示す構造体を用意しています．リンクされるワークエリアは，これらの間に「サンドイッチ」されるわけです．それゆえ，リンクを外す場合でも，特別な処理は一切不要となります．

再入の配慮

今回は，PITを方形波モードで使用することから，再入に関して配慮しないといけません．in_interruptは，そのための変数です．現在割り込みルーチンにいる場合には，この変数をセットします．require_countは，割り込みサービスを要求する回数を記憶します．

割り込みのタイムアップが生じ，再入動作が起こった場合には，require_countのみをインクリメントします．そして，割り込みサービスルーチンの最後でrequire_countをデクリメントし，それが0でなければ再度サービスが必要とされたと判断して，割り込み処理を繰り返します．これによって，割り込み処理がまにあわなかったとしても，必要な回数の割り込み処理が実行されることになります．

single_count/single_workは，シングルイベント用のカウンタとして使用されます．single_workは，プリスケラーです．

タイマ割り込みサービスルーチン

次に，タイマ割り込みのサービスルーチンがあります．まず，データセグメントのセットアップをした後，require_countをインクリメントします．それからEOIを発行し，現在サービス実行中であるかどうかをチェックします．再入動作であればそのままリターンします．そうでなければ，ループ動作を実行します．

ループでは，まずシングルイベントの処理を行います．カウンタのデクリメントを行い，必要ならint7を発行します．そのあとに，マルチイベントのチェックを行います．このプログラムでは，ハイレゾのBIOSのようにワークエリアの並べ変えは行っていません．

ES:BXでワークエリアをポイントします．ポインタを進めた後，そのワ

ワンラインNEWS
任天堂は「スーパーファミコン（米国では「スーパーNES」）」の販売を開始した．日本の半導体業界では，半導体ビジネスとして期待されている．

ークエリアのリンクポインタの値が0であればループを終了します．これは，ポインタがtimer_tailに達したことを意味します．そうでなければ，シングルイベントと同様にカウンタのデクリメントを行います．タイムアップしていない場合には，@continueで@beginの直後からループを繰り返します．タイムアップしたときには，登録された割り込みルーチンを呼び出します．

その後，リピートモードで登録され，かつAHレジスタの戻り値が0でなければ，再び割り込みサービスが有効になるように，ワークエリアの再設定を行います．そうでない場合は，ワークエリアをリンクから外します．このときに，前のワークエリアと後ろのワークエリアをつなぐだけでなく，外すワークエリアのポインタをクリアしておきます．これによって，すでにインアクティブになっているワークエリアを認識します．このような処理は，双方向リンクのほうが簡単に行えます．

最後にrequire_countをデクリメントし，それが0でなければループを繰り返します．0であれば，in_interruptをクリアし，復帰します．

INT1CHのルーチン

次に，タイマ割り込みをサービスするルーチン（INT1CHのルーチン）があります．ハイレゾのBIOSでは，ビープに関するサービスAH=6があるのですが，ここではAH=2～AH=5の4つのサービスのみをサポートしています．これらの範囲でないときには，前の割り込み処理ルーチンを呼び出します．ここで，先程のラベルが有効に効いてきます．

次にレジスタのセーブとデータセグメントの設定を行った後，各サービスに分岐します．AH=2のシングルイベントの場合には，各ワークエリアとINT7の割り込みベクタに値を設定します．これらの値を設定する場合に，あらかじめ割り込みを禁止する必要があります．

AH=3のタイマキャンセルの場合には，まずワークエリアの有効性をチェックします．一度も登録されていないワークエリアの場合には，チェックが有効であるわけではありませんが，リンクされたワークエリアが，割り込みルーチンでリンクから外されたことは認識できます．

AH=4/5の場合には，登録を行います．AH=4はワンショットモード，AH=5の場合にはリピートモードで起動します．これらの違いは，ワークエリアに設定されます．そのほかの各パラメータを設定した後，リンクを構成します．なお，すでにリンクされているワークエリアに再度登録を行ってはいけません．これは単方向リンクでFATALな状態を引き起こします．

init_func

最後にあるinit_funcは，常駐時にRES2から呼び出されるものです．これによって，ハードウェアなどの初期設定を行うことができます．ここでは，PITにクロックに応じた値を設定した後，タイマ割り込みを許可しています．

RES2.ASM

では，もう1つのモジュールRES2.ASMについても説明しておきましょう．この内容の基本的な部分はRESIDENT.ASMと同じです．ですから，異なっている部分についてのみ説明します．

まず，CODEセグメントにEXTRN宣言があります．リンクするモジュールの各変数・ルーチンは，みなCODEセグメントにあるので，ここでEXTRN宣言をしないといけません．その次にint_workの構造体宣言があります．

int2fでは，BX=3/4の割り込み設定のサービスが削除されています．また，リリース時には，interrupt_listをたどって，すべての割り込みを元に戻すようにしてあります．INT2Fについては，これとは別に復元します．

イニシャライズ部分では，-iオプションの処理が削除されています．これに伴い，atohの関数も削除されます．my_idが_TEXTセグメントに入っているので，初期化のところではセグメント操作が必要です．ここでは，毎回DSの値を切り替えています．これに伴い，ASSUMEも繰り返し使用されています．

常駐処理では，INT2Fと同様に，interrupt_listで指定されるすべての割り込みベクタを設定します．もちろん，その前に元の割り込みベクタの内容をワークエリアに退避します．

実際に常駐プログラムを作る場合には，それぞれのプログラムを別々にアセンブルし，最後にリンクします．このときに，RES2のほうをあとにリンクする必要があるのは，RESIDENTの場合と同様です．

テストプログラムTMTEST.C

timerを使用したプログラムをCで記述

このtimerを使った簡単なプログラムをCで書いてみましょう．TMTESTがそれです．ここで重要な

のは，timer_workという構造体です．これはtimerで記述されたものと同様ですが，高級言語が見る必要のない部分は，ダミーとして見えないようになっています．

関数set_timerは，ワークのアドレス，インターラプト処理ルーチンのアドレス，割り込み間隔を引数に取り，INT1CHを発行することによってそれを登録するものです．最初にワークエリアの初期化を行った後，レジスタ変数を利用して値の設定を行い，INT1CHを実行します．

次のrelease_timerはその逆に，リンクを外すものです．同様にレジスタ変数を利用しています．

そのあとに，２つのワークエリアがあり，それぞれに対応する処理ルーチンがあります．これらは画面上のアトリビュートを操作し，割り込みがかかるたびに，リバースアトリビュートを動かしていくだけのものです．func1は最上段を，func2は２段目を動かします．なお，一番左の桁はリバースが残ってしまいます．

mainは，これらを呼び出すだけです．func1は10msごと，func2は50msごとに呼び出すようにしてみます．こうすると，最上段は高速に反転キャラクタが移動し，２段目はゆっくりと移動します．また双方が同時に動作することから，マルチイベントの処理が行われていることがわかります．

これはなにかキーをたたくまで続き，そのあとにrelease_timerでタイマをクリアしてから終了します．

同様なマルチイベントタイママネージャを作成してプログラムで使う場合には，その存在をINT2Fなどでチェックしてから行うようにしたほうが好ましいでしょう．

リスト３

```
#include <stdio.h>
#include <dos.h>

#define INT_ARGS unsigned BP, unsigned DI, unsigned SI, ¥
    unsigned DS, unsigned ES, unsigned DX, unsigned CX, ¥
    unsigned BX, unsigned AX, unsigned PC, unsigned CS, ¥
    unsigned FLAG

typedef struct {
    char dummy[6];

    int count;
    void interrupt (*func)();

    char dummy2[6];
} timer_work;

void set_timer(timer_work *p, void interrupt (*func)(), int count)
{
    p->count = count;
    p->func = func;

    _ES = _DS;
    _BX = (unsigned)p;
    _AH = 5;
//    _AH = 4;
    geninterrupt(0x1c);
}

void release_timer(timer_work *p)
{
    _ES = _DS;
    _BX = (unsigned)p;
    _AH = 3;
    geninterrupt(0x1c);
}

timer_work timer_1, timer_2;

void interrupt func1(INT_ARGS)
{
    static int i = 0;
    char far *p = MK_FP(0xa200, 0);

    p[i] ^= 0x4;
    i += 2;
    if (i >= 160)
        i = 0;
    p[i] ^= 0x4;

    &AX;
    AX = 0xff00;
}

void interrupt func2(INT_ARGS)
{
    static int i = 0;
    char far *p = MK_FP(0xa200, 160);

    p[i] ^= 0x4;
    i += 2;
    if (i >= 160)
        i = 0;
    p[i] ^= 0x4;

    &AX;
    AX = 0xff00;
}

main()
{
    set_timer(&timer_1, func1, 1);
    set_timer(&timer_2, func2, 5);

    _AH = 0;
    geninterrupt(0x18);

    release_timer(&timer_1);
    release_timer(&timer_2);
}

```


ワンラインNEWS
日立製作所は低電圧でも高速動作が可能な４ビットMCUを製品化，サンプル出荷を開始した．2.7ボルトの電圧での最小命令実行時間は１マイクロ秒を誇る．

PRODUCTS REVIEW

元祖ダートマス大学版の直系
日本語TrueBASIC V3.0J

パーソナルコンピュータの草創期において，BASICのはたした役割は大きなものでした．というより，機械語レベルのプログラムのほかはBASIC以外の言語がなかったのです．MS-DOSが標準的な位置を占めるに従って，BASICの存在はC言語に圧迫されるようになったのですが，近年のQuickBASICの成功を見てもわかるように，いまだに根強い人気を保っています．このほど，BASICの生みの親であるケメニーとカーツによるTure BASICが日本語化され，(株)クレオより発売されました．ここでは，その概要をレポートします．

なぜ，True(本物)なのか？

BASICは，1960年，ダートマス大学のジョン.G.ケメニーとトーマス.E.カーツの両博士によって開発されました．彼らは，この系譜を引くBASICをTrueBASIC（本物のBASIC，または両博士のイニシャルをとってKK-BASIC）と呼び，それ以外のさまざまなBASICを傍流として位置づけ，「ストリートBASIC」と呼んでいます．現在のTrueBASICは，両博士により1984年に構造化言語として再生されたものです．従来のマシン依存の制約から解放され，きわめて高い自由度を備えた言語システムです．

特徴としては，次のような点があげられます．

- ANSI規格に準拠し，シンプルな命令体系により，わかりやすいプログラミングが可能．
- 構造化プログラミングに有効な制御文を多数用意している．
- プログラミングから実行までが可能な，統合環境をサポートしている．
- プログラムに行番号を必要としないため，モジュール化が可能．
- 行列演算機能を標準でサポート．
- 数値演算コプロセッサに対応し，より高速かつ高精度な演算が可能．
- UNIX，Macintosh，IBM PCなどで利用可能．

開発者によるリファレンスを添付

TrueBASICのパッケージには，プログラムディスク1枚とユーザー登録カード，4分冊のマニュアル（セットアップマニュアル，TrueBASICユーザーズガイド，

TrueBASICリファレンスマニュアル，TrueBASICクイックリファレンス）が付属しています．

なかでも，『TrueBASICリファレンスマニュアル』（啓学出版）は，TrueBASICの設計者が書き下ろしたTrueBASICの辞典で，あえて市販本を添付してあるだけあって，マニュアルにありがちな単調な技術解説を超えた内容になっています．

また，『TrueBASICユーザーズガイド』は，初級，中級，上級編の3部構成となっており，付属のサンプルプログラムを利用してわかりやすく解説されています．

付録では，プログラム内で扱うことのできる5種類のファイルフォーマット（バイトファイル，固定長ファイル，可変長ファイル，ストリームファイル，テキストファイル）の構造や，イメージ文字列の構造などが詳細に解説されています．

インストール

TrueBASICシステムをインストールするには，付属のセットアップユーティリティを使います．セットアップユーティリティは，ハードディスクやフロッピーディスク（2ドライブ），98NOTEのRAMドライブにも対応しています．インストール作業が終了すると，ドライブのルートディレクトリにTRUEディレクトリが作成され，さらにその中にRUNTIME，TBDO，TBHELP，TBLIBSの各サブディレクトリが作成されて必要なファイルが保管されます．各ディレクトリの構成は図1のようになります．

TrueBASICの操作環境

TrueBASICを起動するには，コマンドプロンプトから「HELLO」と入力します．すると，画面1のようなTrueBASICのエディタ画面が表示されます．

TrueBASICのエディタでは，ファンクションキーを使う方法，キーボードから直接コマンドを入力する方法，メニューを選択する方法（マウス対応）の3通りの方法を使って作業を進めることができます．もちろん，日本語フロントエンドプロセッサを組み込めば，日本語を入力することができます．

エディタ画面には，編集メニュー，カスタムメニュー，実行メニュー，オプションメニューの5種類のメニューが表示されています．マウスが装備されている場合には，マウスでメニューを選択して作業をすすめると効率的です．各メニューに配置されているコマンドを選択して作業を進めることになります．メニューのコマンドは，一部をのぞいてキー入力のコマンドにも対応しています．また，ファンクションキーを使って操作することもできます．

キーボードから直接コマンドを入力する方法では，表1のコマンドを利用することができます．TrueBASICのコマンドは，コマンドウインドウの中のコマンド行に入力し，TrueBASICのプログラムの中では使用することはできません．

図1　セットアップ後のディレクトリの構成

```
¥ ── TRUE(システム本体、ドライバ、サンプルなど)
          ├── RUNTIME（EXEファイル作成ツール）
          ├── TBDO（プリプロセッサ）
          ├── TBHELP（ヘルプファイル）
          └── TBLIBS（ライブラリ）
```

プログラムの作成手順

ファイルの種類

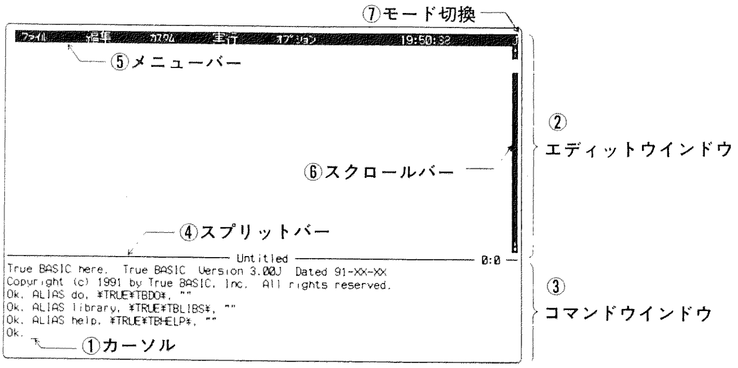
TrueBASICで使用するファイルは，表2のように6種類に分類されます．

ファイル名の規則

TrueBASICのエディタでは，作成したプログラムファイルに自動的に拡張子「TRU」がつけられます．また，プログラムファイルをコンパイルすると，拡張子は自動的に「TRC」になります．プログラムファイル，コンパイル済みのファイルともに任意の拡張子

ワンラインNEWS
日本DGはスーパーミニコン「ECLIPSE MVシリーズ」の中位3機種を発売した．従来機種に比べ40%性能を強化したのが特徴

画面1　TrueBASICのエディタ画面（起動画面）



①カーソル
キーボードから直接コマンドを入力する場合，この位置に表示される．
②エディットウィンドウ
プログラムの入力や編集作業に使用．
③コマンドウィンドウ
コマンドの入力に使用．実行結果やエラーメッセージも表示．50行までのバックスクロールが可能．
④スプリットバー
この位置を変更して各ウインドウの大きさの変更が可能．カレントプログラム名（Untitled），カーソルの位置（0：0）が行数と桁数で表示される．
⑤メニューバー
5種類のメニュータイトルとシステムが管理している時刻を表示．
⑥スクロールバー
マウスを使って，長いプログラムの中の必要な場所に移動するのに使用．
⑦モード切換
JまたはEの文字で表示．Jの場合は，メニューバー，ファンクションキー，メッセージが日本語表示され，Eでは英語表示される．マウスでクリックして切り換える．

を付けて保存することもできます．

プログラムの作成

プログラムの作成は，図2の手順で行います．

高度な機能

TrueBASICでは，以下のような高度な機能もサポートされています．

・ヘルプファイルのカスタマイズ

TBHELPディレクトリ内に保管されている各ヘルプファイルは，すべてテキストファイルで作成されているため，記述の内容の変更や独自のヘルプファイルを作成し追加することができます．

・SHELL機能

TBLIBSディレクトリ内のSHELL.TRUファイルには，Dosサブルーチンとshellサブルーチンが含まれており，この2つのサブルーチンを使うとTrueBASICを起動中に，直接モードまたはプログラム内からにかかわらず，MS-DOSのコマンドを利用することができます．

・PEEKとPOKEの利用

PEEKとPOKEを使うと，8086のフ

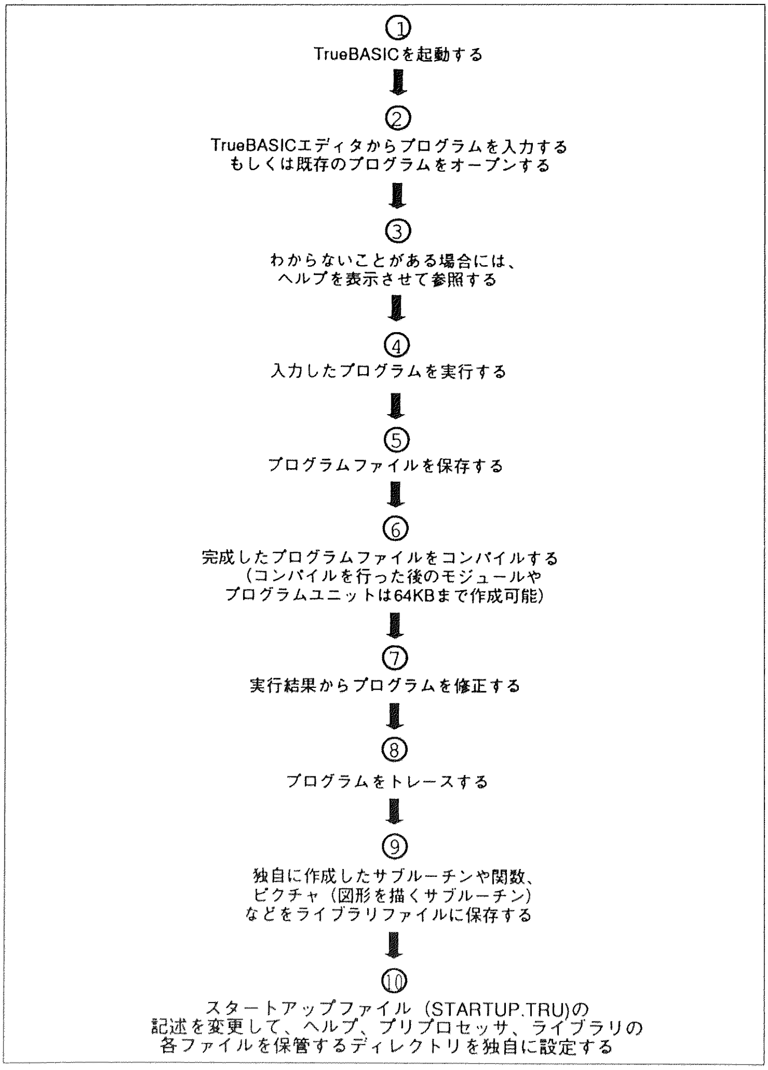
表1　TrueBASICコマンド一覧

ALIAS	BREAK	BYE	CD
CHANGE	CLOSE	COLORS	COMPILE
CONTINUE	COPY	DELETE	DO
ECHO	EDIT	ENTER	FILES
FIND	FORGET	HELP	INCLUDE
INFO	KEEP	KEY	LANGUAGE
LIST	LOAD	LOCATE	MARK
MENUS	MODE	MOVE	NEW
NOLET	OLD	RENAME	REPLACE
RUN	SAVE	SCRIPT	SPLIT
SWITCH	TO	TRY	UNSAVE
VIEW KEYS			

表 2　TrueBASICで使うファイルの種類

ファイル	働き
プログラムファイル	TrueBASICのエディタもしくはMS-DOSのテキストエディタを使って作成された，TrueBASICで実行することのできるプログラムファイル.
ライブラリファイル	サブルーチン，関数などをまとめて保存してあるファイル. ライブラリファイルとして保存しておくことにより，いつでも呼び出して使用することができる.
ヘルプファイル	ヘルプ情報の記載されたテキストファイル.
プリプロセッサファイル	カレントプログラムを直接変更する処理を行うファイル. DOコマンドまたはカスタムメニューなどを使って実行する.
スクリプトファイル	TrueBASICのコマンドを列挙したファイル. スクリプトファイルを使うと列挙されたコマンドが自動的に実行される.

図 2　プログラム作成の手順



ラットアドレスを使用して，メモリのバイト(I/Oポートのバイトも可能)を点検したり，変更することができます.

・MS-DOSのコマンドラインのパラメータを引き渡す

PROGRAM文を使うことによって，MS-DOSのコマンドラインから特定のプログラムへパラメータを引き渡すことができます. これによってプログラムは，MS-DOSのコマンドラインのパラメータをメインプログラムの実行可能な最初の行に受け取ることができるようになります.

・ツールキットの利用

目的に応じて，以下のような別売のツールキットを利用することができます. ツールキットのパッケージは，順次発売されていく予定です.

①開発支援ツールキット
②並べ替え・検索用ツールキット
③データ入力フォーム管理ツールキット
④3次元グラフィックスツールキット
⑤ビジネス用グラフィックスツールキット
⑥科学技術用グラフィックスツールキット
⑦統計用グラフィックスツールキット
⑧数学者用開発支援ツールキット
⑨通信サポート用ツールキット
⑩文字列操作ツールキット

■日本語TrueBASIC V3.0J

本体：PC-9800シリーズ／メモリ：640KB以上／ディスク装置：1MBタイプ2ドライブ以上(98NOTEのRAMドライブ対応）／MS-DOS：MS-DOS 2.11以上／価格：18,000円／株式会社クレオ 電話03-3456-4650

ワンラインNEWS
東京ガスは小型の車載型設備管理システムを開発. ペン入力パソコンを組み込み, 従来は後部荷台を占有する大きさだったのが助手席の前面に収容できる大きさにした.

情報処理技術者試験

第２種合格！トレーニング

第８回　COBOL

静岡情報ビジネス専門学校　白井儀巳・佐野有一

COBOL の出題傾向

過去５回の COBOL に関する出題傾向は表１の通りです．表からもわかる通り，表操作とコントロールブレイク処理は毎回必ずといってよいほど出題されています．今月はこれらのテーマを重点的に解説します．

●表１　COBOL に関する出題傾向

出題年度		出題テーマ	出題内容
昭和63年春	問５ 問９ 問13	得意先別請求書の作成 売れ筋商品一覧表の作成 平均気温棒グラフの作成	コントロールブレイク 表操作，２次元表検索 表操作
昭和63年秋	問５ 問９ 問13	支店別売上高の表示処理 都道府県別家計支出構成リストの作成 農業後継者の抽出，整列処理	表操作，索引ファイル コントロールブレイク仕様変更による修正，表操作 表操作，SEARCH，SORT
平成元年春	問５ 問９ 問13	売上高に応じたクラス分け処理 都道府県別，産業分類別の年間販売額一覧表の作成 市町村別山林面積，戸数の集計処理	表操作，SEARCH 表操作，２次元表集計 コントロールブレイク，表操作，EVALUATE
平成元年秋	問５ 問９ 問13	課別平均残業時間レポートの作成 作物出荷量の上位市町村の抽出処理 作業日報のエラーチェック処理 表操作	コントロールブレイク コントロールブレイク，表操作，索引ファイル
平成２年春	問５ 問９ 問13	被扶養者の抽出一覧表の作成 仕入額と販売額の一覧表の作成 検索条件該当社員の抽出表示	表操作 コントロールブレイク マッチング 表操作

表操作

■表の概念

日常生活においては，列車の運賃表や保険料金表のようにある目的のために同一形式のデータが順序よく配置された表をよく見かけます．COBOL においても同一形式のデータが複数連続している場合には，個々のデータごとに定義せずに複数のデータを１つにまとめて定義します．

■表の確保

表はデータ部で必要とする要素の数だけ OCCURS 句で指定して定義します．たとえば，表の名前を TBL とし PIC 9(5)の属性をもつ領域を12個確保するには図１のように記述します．

●図１

```
01   TBL.
  02   TUKI   PIC 9(5)   OCCURS  12.
```

TBL	TUKI(1)	TUKI(2)	……	TUKI(12)

図１のように一方向へデータ領域が確保された表を１次元の表と呼び，図２のように２方向へデータ領域が確保された表を２次元の表と呼びます．

●図２

```
01   TBL.
  02   NEN   OCCURS  3.
    03   TUKI   PIC 9(5)   OCCURS  12.
```

TUKI(1,1)	TUKI(1,2)	……	TUKI(1,12)
TUKI(2,1)	TUKI(2,2)	……	TUKI(2,12)
TUKI(3,1)	TUKI(3,2)	……	TUKI(3,12)

■表の要素

表を構成する１つ１つのデータ項目を表の要素と呼びます．表の中での要素の位置は，並び順での番号で指定します．たとえば，図１で２番目の要素を参照するには TUKI(2) と指定し図

2で2行目の3番目の要素を参照するにはTUKI(2,3)と指定します．この要素の位置を指定するためのカッコの中の数字を添字と呼びます．

■表探索

表探索専用の命令として，COBOLにはSEARCH文とSEARCH ALL文があります．

●逐次探索　SEARCH文

SEARCH文は指定した条件と一致する要素を表の中から順次探索します．

● SEARCH文の留意点

①一意名1はOCCURS句とINDEXED BY句で定義されていなければならない．

②探索を開始する表の要素の位置をSET文により指標に設定しなければならない．

③条件一致しない場合は自動的に指標の値が1つ加算される．

④指標が1つである場合にはVARYING句は省略することができる．

●2分探索　SEARCH ALL文

SEARCH ALL文は指定した条件と一致する要素を表の中から2分探索します．

● SEARCH ALL文の留意点

①表の要素の内容があらかじめ小から大または大から小に並んでいなければならない．

②条件式は等価(=)でなければならない．

③指標が1つである場合にはVARYING句は省略できる．

[OCCURS 句の一般形式]

```
OCCURS 整数 TIMES
[{ASCENDING }  KEY IS {データ名} ……]
[{DESCENDING}
 [INDEXED BY {指標名} …… ]
```

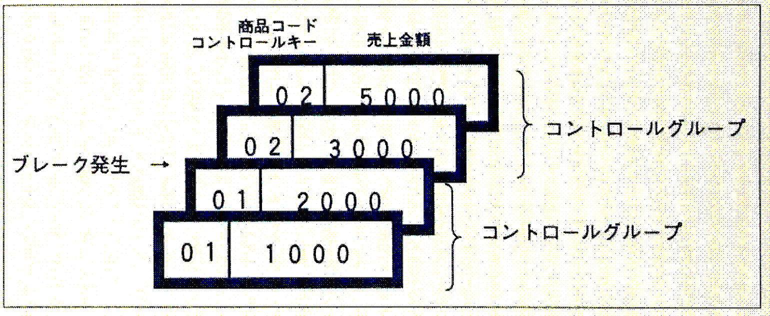
[SEARCH 文の一般形式]

```
SEARCH 一意名1 [VARYING {一意名2}]
                        {指標名 }
  [AT END 無条件文1]
  {WHEN 条件 {無条件文2         }} ……
             {NEXT SENTENCE     }
  [END-SEARCH]
```

[SEARCH ALL 文の一般形式]

```
SEARCH ALL 一意名1 [AT END 無条件文1]
     {データ名1 {IS EQUAL TO} {一意名3 }}
WHEN {          {IS =       } {定数1   }}
     {                        {算術式1 }}
     {条件名1                           }
  [     {データ名2 {IS EQUAL TO} {一意名4 }}]
  [AND  {          {IS =       } {定数2   }}]
  [     {                        {算術式2 }}]
  [     {条件名2                           }]
  {無条件文2    }
  {NEXT SENTENCE}
  [END-SEARCH]
```

●図 3



コントロールブレイク処理

■コントロールブレイク処理とは

事務処理には小計，大計，総計の集計処理がよく行われます．このような処理をコントロールブレイク処理（グループ集計）と呼びます．それぞれのグループ処理の制御をする項目をコントロールキーと呼び，集計処理を行うためには，このキーが順序よく並んでいなければなりません．集計の対象となるグループのコントロールキーが同じならば集計処理を行い，異なっているならば結果を出力します．

ブレイクが発生した場合には，次のような点に留意してください．

①ブレイクが発生したグループの集計結果を出力し，1つ上位の集計項目に加算し，出力した集計項目をゼロにする．

②新しいグループの集計処理のためにコントロールキーを入れ替える．

■基本的な集計処理のフローチャート

大キー，小キーをコントロールキーとして小計，大計，総計を出力します．なお，キーは小から大に順番にならんでおり，複数のレコードが存在するものとします．

①ファイルが終了するまで総計処理を繰り返す．

②大計をゼロにして大キーを保存域に退避させる．

③ファイルが終了するまでまたは大キーと保存キーとが異なるまで大計処理を繰り返す．

④小計をゼロにして小キーを保存域に退避させる．

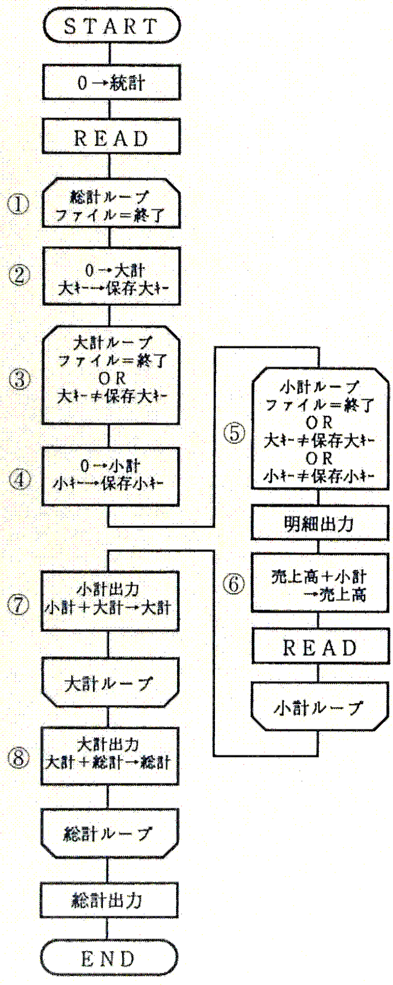
⑤ファイルが終了するまでまたは大キーと保存キーが異なるまで，もしくは小キーと保存キーが異なるまで小計処理を繰り返す．

⑥同じグループの小計処理を行う．

⑦小計を出力し小計を大計に加算する．

⑧大計を出力し大計を総計に加算する．

●図 4



レコードレイアウト

大キー	小キー	売上高

EXAMPLE

次のCOBOLプログラム中の［　　］を埋めてプログラムを完成せよ．

［プログラムの説明］

商品ファイルを入力し，売上一覧表を作成するプログラムである．

(1)　商品ファイルのレコード様式は，次の通りである．

商品コード 3けた	売上数量 5けた

(2)　商品コードの数は100である．

(3)　プログラムは，商品単価を求めるために次の表をもっている．

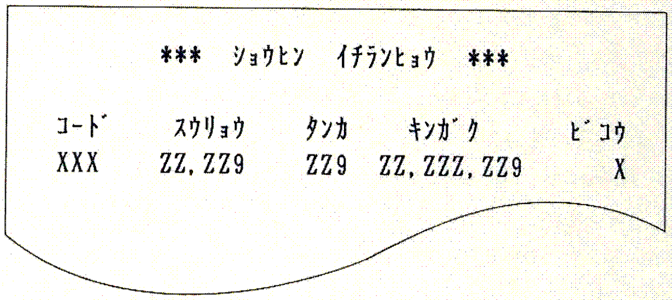
商品コード	商品単価
〃	〃
⋮	⋮
〃	〃

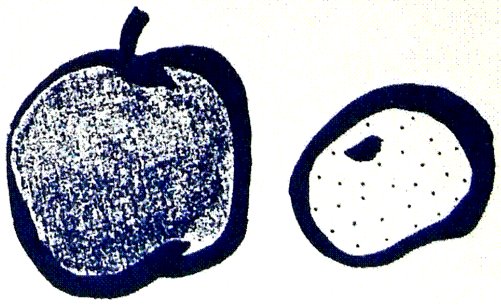
（100）

(4)　TABSETという名前のプログラムを呼ぶと上記表の商品コード欄に商品コードが，また，商品単価欄に商品単価がセットされる．

(5)　売上一覧表の様式は図のとおりであり，商品コードが表の中に存在しないときには，備考欄に＊印を印字し，単価0で計算する．なお，1ペ

●売上一覧表の様式





```
DATA            DIVISION.
FILE            SECTION.
FD  SYOHIN-F.
01  SYOHIN-R.
    02  SYOHIN-CODE   PIC X(3).
    02  SYOHIN-SU     PIC 9(5).
FD  PRINT-F.
01  PRINT-R           PIC X(132).
WORKING-STORAGE   SECTION.
01  TANKA-TAB.
    02  T-T OCCURS 100 INDEXED BY I
            ASCENDING KEY T-CODE.
        03  T-CODE    PIC X(03).
        03  T-TANKA   PIC 9(03).
01  W-TANKA          PIC 9(03).
01  L-CNT            PIC 9(02) VALUE 30.
01  EOF              PIC 9(01) VALUE  0.
01  MID1.
    02               PIC X(20) VALUE SPACE.
    02               PIC X(21)
        VALUE "*** ｼｮｳﾋﾝ ｲﾁﾗﾝﾋｮｳ ***".
  01  MID2.
    02               PIC X(05) VALUE SPACE.
    02               PIC X(20)
        VALUE "ｺｰﾄﾞ   ｽｳﾘｮｳ    ﾀﾝｶ ".
    02               PIC X(20)
        VALUE "   ｷﾝｶﾞｸ      ﾋﾞｺｳ   ".
01  MEISAI.
    02               PIC X(10) VALUE SPACE.
    02  M-CODE       PIC X(03).
    02               PIC X(03) VALUE SPACE.
    02  M-SU         PIC ZZ,ZZ9.
    02               PIC X(04) VALUE SPACE.
    02  M-TANKA      PIC ZZ9.
    02               PIC X(02) VALUE SPACE.
    02  M-KIN        PIC ZZ,ZZZ,ZZ9.
    02               PIC X(05) VALUE SPACE.
    02  M-BIKOU  PIC X(01).
PROCEDURE         DIVISION.
INIT-RTN.
    OPEN INPUT SYOHIN-F OUTPUT PRINT-F
    CALL "TABSET" USING TANKA-TAB.
MAIN-RTN.
    READ SYOHIN-F AT END MOVE 1 TO EOF
    END-READ
    [          (1)          ]
END-RTN.
    CLOSE SYOHIN-F PRINT-F
    STOP RUN.
SYORI-RTN.
    SEARCH ALL T-T
       AT END
            MOVE 0 TO W-TANKA
            MOVE "*" TO M-BIKOU
    [          (2)          ]
            MOVE SPACE TO M-BIKOU
            MOVE T-TANKA (I) TO W-TANKA
    END-SEARCH
    [          (3)          ]
    MOVE SYOHIN-CODE TO M-CODE
    MOVE SYOHIN-SU    TO M-SU
    MOVE W-TANKA      TO M-TANKA
    PERFORM MID-RTN
    WRITE PRINT-R FROM MEISAI AFTER 1
    COMPUTE L-CNT = L-CNT + 1
    READ SYOHIN-F AT END MOVE 1 TO EOF
    END-READ.
MID-RTN.
    [          (4)          ]
      WRITE PRINT-R FROM MID1 AFTER PAGE
      WRITE PRINT-R FROM MID2 AFTER 2
      MOVE 0 TO L-CNT
    END-IF.
```

ージの明細行は30行とし，改ページを行う．

解　説

(1)　このプログラム中で，主処理が行われているSYORI-RTNを実行する命令がないことに気がつけば，外PERFORMを使うことがわかる．どのような条件で何回繰り返せばよいかは，問題を読めばわかる通りファイルが終了するまでである．そこで，READ文のAT END以降の無条件命令をみると，EOFに1を移しているので，条件はそれを＝で結ぶだけである．

(2)　SEARCH命令は，検索したい項目と，テーブル中の項目とを比較し，条件が満たされれば，なんらかの処理を行う命令であるが，そのための条件式がSEARCH命令の中に存在しない．しかもSEARCH ALLの場合，等しい(=)という条件式しか使えないことに気をつける．

(3)　編集項目に計算結果を直接入れるこのようなやり方は，よく出題されている．

(4)　改ページの基本的なやり方である．このような問題については，ライン・カウントの初期値やタイミングについてトレースする力を養うことが大事である．

解　答

(1)　PERFORM　SYORI−RTN UNTIL EOF ＝ 1

(2)　WHEN　T−CODE (I) ＝ SYOHIN−CODE

(3)　COMPUTE M−KIN ＝ W−TANKA ＊ SYOHIN−SU

(4)　IF L−CNT >= 30
　　またはIF L−CNT ＝ 30

EXERCISE

次のCOBOLプログラム中の［　　］を埋めてプログラムを完成せよ．

［プログラムの説明］

収支ファイルを入力し，会社別収支一覧表を作成するプログラムである．

(1)　収支ファイルのレコード様式は，次の通りで，会社コードの昇順に並べられている．

会社コード 3けた	商品コード 4けた	収入 6けた	支出 6けた

(2)　会社コードの数は50である．

(3)　プログラムは，会社名を求めるために次の表をもっている．

50

会社コード	会社名
〃	〃
⋮	⋮
〃	〃

(4)　TABSETという名前のプログラムを呼ぶと上記表の会社コード欄に会社コードが，また，会社名欄に会社名がセットされる．

(5)　会社別収支一覧表の様式は次のとおりであり，会社コードが表の中に存在しない時には，画面に会社コードと共にエラー表示をし，データを読みとばす．

(6)　1ページの明細行は30行とし改ページを行い，会社コードが変った時には，会社合計を出力し改ページする．

解　答

(1)　T-K OCCURS 50 INDEXED BY IND

(2)　SET IND TO 1

(3)　MOVE T-NAME (IND) TO M-NAME

(4)　COMPUTE L-CNT = L-CNT + 1

(5)　MOVE " ｺﾞｳｹｲ " TO M-NAME

●会社別収支一覧表の様式

```
        ***  ｶｲｼｬﾍﾞﾂ  ｼｭｳｼ  ｲﾁﾗﾝﾋｮｳ  ***

ｶｲｼｬﾒｲ   ｼｮｳﾋﾝｺｰﾄﾞ      ｼｭｳﾆｭｳ          ｼｼｭﾂ
Aｶｲｼｬ      0001        350,000        240,000
Aｶｲｼｬ      0002        485,000        524,000
Aｶｲｼｬ      0003        229,000        115,000
ｺﾞｳｹｲ                1,064,000        879,000
--------------------------------------------------
        ***  ｶｲｼｬﾍﾞﾂ  ｼｭｳｼ  ｲﾁﾗﾝﾋｮｳ  ***

ｶｲｼｬﾒｲ   ｼｮｳﾋﾝｺｰﾄﾞ      ｼｭｳﾆｭｳ          ｼｼｭﾂ
Bｶｲｼｬ      1001        850,000        364,000
```

```
 DATA                   DIVISION.
 FILE                   SECTION.
 FD  K-FILE.
 01  K-REC.
     02  K-CODE   PIC X(03).
     02  K-S-CODE PIC X(04).
     02  K-IN     PIC 9(06).
     02  K-OUT    PIC 9(06).
 FD  O-FILE.
 01  O-REC        PIC X(132).
 WORKING-STORAGE   SECTION.
 01  KAISYA-TAB.
     02 [          (1)          ]
         03  T-CODE     PIC X(03).
         03  T-NAME     PIC X(06).
 01  L-CNT          PIC 9(02) VALUE 90.
 01  EOF            PIC 9(01) VALUE  0.
 01  W-IN           PIC 9(08).
 01  W-OUT          PIC 9(08).
 01  W-CODE         PIC X(03).
 01  MID1.
     02             PIC X(20) VALUE SPACE.
     02             PIC X(20)
         VALUE "*** ｶｲｼｬﾍﾞﾂ ｼｭｳｼ ｲﾁﾗ".
     02             PIC X(20)
         VALUE "ﾝﾋｮｳ ***            ".
 01  MID2.
     02             PIC X(05) VALUE SPACE.
     02             PIC X(20)
         VALUE "ｶｲｼｬﾒｲ ｼｮｳﾋﾝｺｰﾄﾞ     ".
     02             PIC X(20)
         VALUE " ｼｭｳﾆｭｳ        ｼｼｭﾂ  ".
 01  MEISAI.
     02             PIC X(05) VALUE SPACE.
     02  M-NAME     PIC X(06).
     02             PIC X(03) VALUE SPACE.
     02  M-S-CODE   PIC X(04).
     02             PIC X(03) VALUE SPACE.
     02  M-IN       PIC ZZ,ZZZ,ZZ9.
     02             PIC X(01) VALUE SPACE.
     02  M-OUT      PIC ZZ,ZZZ,ZZ9.
 PROCEDURE              DIVISION.
 INIT-RTN.
     OPEN INPUT K-FILE OUTPUT O-FILE
     CALL "TABSET" USING KAISYA-TAB
     PERFORM READ-RTN.
 MAIN-RTN.
     PERFORM UNTIL EOF = 1
     [               (2)               ]
     SEARCH T-K
       AT END
         DISPLAY "ｺｰﾄﾞ ﾅｼ :" K-CODE
         PERFORM READ-RTN
       WHEN T-CODE (IND) = K-CODE
     MOVE K-CODE TO W-CODE
     MOVE 0      TO W-IN W-OUT
     PERFORM UNTIL K-CODE NOT = W-CODE
              OR EOF = 1
         MOVE K-S-CODE TO M-S-CODE
         MOVE K-IN     TO M-IN
         MOVE K-OUT    TO M-OUT
     [               (3)               ]
         COMPUTE W-IN  = W-IN  + K-IN
         COMPUTE W-OUT = W-OUT + K-OUT
         IF  L-CNT >= 30
             PERFORM MID-RTN
         END-IF
         WRITE O-REC FROM MEISAI AFTER 1
     [               (4)               ]
         PERFORM READ-RTN
       END-PERFORM
     [               (5)               ]
       MOVE SPACE    TO M-S-CODE
       MOVE W-IN     TO M-IN
       MOVE W-OUT    TO M-OUT
       WRITE O-REC FROM MEISAI AFTER 3
       MOVE  99    TO   L-CNT
     END-SEARCH
     END-PERFORM
     CLOSE K-FILE O-FILE
     STOP RUN.
 MID-RTN.
     WRITE O-REC FROM MID1 AFTER PAGE
     WRITE O-REC FROM MID2 AFTER 2
     MOVE 0 TO L-CNT.
 READ-RTN.
     READ K-FILE AT END MOVE 1 TO EOF
     END-READ.
```

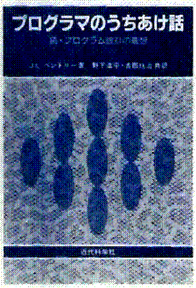
Book Review

■中村正三郎

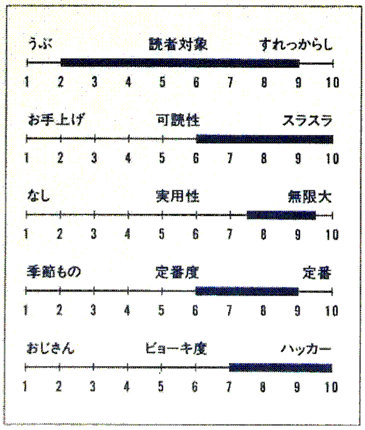
ユーモアあふれる文章でつづられる磨き上げられたアイデアやコツ

■プログラマのうちあけ話

J. L. ベントリー著
野下浩平・古郡廷治訳
近代科学社　3000円



■評価チャート



米国計算機学会の機関誌CACM (Communications of the ACM)には，絶大な支持を受けている名物コラムがあった．コラムニストが，プログラミングの現場で出会った数々の出来事から，さまざまな有用なアイデア，教訓をユーモアあふれる文章でつづるコラムである．しかも，そこで述べられるアイデア，教訓は，決して机上の空論ではない．多くのプログラマが，プログラミングを通じて磨き上げてきたアイデア，プログラマなら知っておきたいプログラミング上のちょっとしたコツなどがちりばめられているのだ．

このコラムこそ，本書の元となった，"Programming Pearls"であり，そのコラムニトとは，本書の著者，J. L. ベントリーである．

本書には，前作があり，その原題が元のコラムと同じ，Programming Pearlsである．これはJ. L. ベントリー著，野下浩平訳『プログラム設計の着想』(近代科学社)として邦訳がすでに出ている．これも得るところが多いと思うので，一読をお薦めする．

本書は，その続編であり（といっても内容面ではかなり違うが），原題More Programming Pearlsの名に恥じない，タメになることが集めてある．

15のコラムが集めてあるが，評者がとくに面白く読んだのは，次のものである．

コラム1　プロファイラ：評者は昔，この種のことをやっていたから，パソコンのプログラマも，もっとプロファイラを使って，プログラムを計測して効率向上を目指すとよいのにと思っている．

コラム5　ゴルディオスの結び：問題をどう解くかの話．先入観に縛られなければ，思わぬ解答を得ることができる．パズル的要素もある示唆に富んだ内容である．このコラムの演習問題3は，解答を見たときあっけにとられた．いかに先入観にとらわれていたか，愕然とさせられた．

このように本書には，各コラムごとに演習問題がついており，巻末に多くの解答がついているので，大変に親切である．

コラム10　文書の設計：プログラマだけではなく，ワープロを使っている多くの人に読んでもらいたい一編．見た目にも読みやすく美しい文書を書くためのヒントがちりばめられている．

コラム13　見事なサンプル：アルゴリズムがどう練られていくか．たしかに見事なサンプルがここにある．取り上げられるアルゴリズムは，ランダムなサンプルを生成するものである．

プログラムの多くは，Awkで書いてある．CとAwkの簡単な説明は付録にあり，Awkで書かれたライブラリルーチンも付録にソースが載っている．

前作の邦題は，少々お堅いものだったから，本書の題名が，少しくだけて「うちあけ話」となったのは，ベントリーのユーモアを交えた軽妙な筆致を思い起こさせて，好感がもてる．ハードカバーだったのが，今回はソフトカバーになった．題名の変更とあいまって，

ワンラインNEWS
三菱電機は4Mビット DRAMを多ビット製品2種の量産を開始した．多ビット製品の投入でパソコンなどが1Mから4Mへの置き換え需要拡大を見込む．

読みやすく感じるのは私だけであろうか．

プログラマ諸氏は，一度この２冊を手に取ってみてください．しばらくは楽しい時間が過ごせるでしょう．

内外のWindows関連書が続々登場！！

古田島義和（神田・書泉グランデ）

最近，Windows 3.0の訳本・書き下ろし図書が急増しています．その中でも，翔永社の「Windows 3.0特別集中講義」が和書の月刊ベスト10で第５位と健闘しています．すでに出版されているアスキー出版の「プログラミングWindows Vol.1,2」は，Microsoft Pressの訳本で，ユーザーから絶大な信頼を受けているC. Petzold氏の著書で，こちらはロングセラーになっています．もう１点，同社から「Windows3ブック」（L. Lovenz，O'Mara共著）が出ていますが，こちらはいまひとつ伸び悩みという感じです．また，同社のおはこであるラーニングシステムシリーズで「入門MS-Windows」が出ました．全３部作で，ビギナーも手軽に使える解説書です．ほかに書き下ろしは，エーアイ出版の「入門MS-Windows Ver.3.0」やナツメ社の「MS-Windows 3.0ハンドブック」が発売以来よく売れています．そのほか，ナツメ社の「まるごとガイドWindows 3.0」，「これなら使えるWindows 3.0」，ソフトバンクの「MS-Windowsのおもちゃ箱」，技術評論社の「やさしくわかるMS-Windows 3.0」，エーアイ出版の「MS-Windows Ver.3.0環境設定編」，技法堂出版の「入門日本語Windows 3.0」など，それぞれ特徴を出して出版されていますが，やはり訳本が人気の中心のようです．

富士ソフトウェアの「ピーター・ノートンWindows 3.0パワープログラミング（上巻・下巻）」が出版．これはBantamの「Windows 3.0 Programming Techniques」の訳本で，ユーザーの期待の現れか，発売の週にはベスト10の第１位になりました．また，アスキー出版から「プログラミングWindows Version3」が出版されました．これは先ほど紹介したMicrosoft Press社の人気スター（？），C. Petzold氏の「Programming Windows 2/E」の訳本で，洋書も入荷するたびにすぐ売り切れになるほどの勢いです．米国のDr. Dobbs Journal誌でも，同作品を信頼できる一冊と称賛していました．ほかに書評として，Addison Wesley「Windows 3.0 for BASIC Programmer's」，M&T「Windows 3:A Developer's Guide」，Bantam「Windows 3 Power Tools」の３冊が取り上げられていましたが，いずれも日本のユーザーに支持されているものばかりで，順調な売れ行きを示しています．

新刊では，オブジェクト指向の洋書が今年暮れから来年春ころまでに入荷します．その中でもカリフォルニア大学サンタバーバラ校で開催されている「Technology of Object Oriented Languages and Systems」（略してTOOLS）をまとめた本がPrentice Hallから「TOOLS4」として出版されます．TOOLSは，オブジェクト指向の第一人者であるB. Meyer氏が議長を務めているので，興味深い図書です．

そのほかの新刊として，いま，啓学出版，アスキー出版などで訳本になって話題をふりまいているのが，O'Reilly社の本です．「Basic System Administration」（Frisch著）は，UNIXシステム管理をより理解するのに最適．「Learning GNU Emacs」（D. Cameron，B. Rosenblatt著）は洋書で学習ものとしては，日本初の入荷になります．共立出版，啓学出版からの訳本が好評なので，この新刊も大いに期待したいものです．

■和書&洋書ベスト10　（書泉グランデ調べ：1991年8月1日～8月31日）

1	UNIX Cプログラミング	アスキー	1	C++ Programming Language 2/E	Addison Wesley
2	UNIXシステムチューニング	アスキー	2	Undocumented DOS Ver.5.0	Addison Wesley
3	インサイドGNU Cコンパイラ	啓学出版	3	C++ Primer 2/E	Addison Wesley
4	コンピュータ・ネットワーク	岩波書店	4	Object-Oriented Modelling & Design	Prentice Hall
5	Windows 3.0特別集中講義	翔泳社	5	Compilers (paper)	Addison Wesley
6	UNIX4.3BSDの設計と実装	丸善	6	Programming Perl	O'Reilly
7	電脳騒乱節 Vol.2	技術評論社	7	ResEdit Complete	Addison Wesley
8	C98スーパーライブラリ	ソフトバンク	8	MS-DOS Programmer's Reference	Microsoft Press
9	電脳騒乱節 Vol.1	技術評論社	9	Design & Implementation of 4.3 BSD UNIX Operationg System Answer Book	Addison Wesley
10	WTERMで今日から始めるパソコン通信	秀和システム	10	Exploring Mathematics & Mathematica	Addison Wesley

ワンラインNEWS
台湾や香港など東南アジアで半導体メモリの販売価格が大幅に下がった．米国を中心としたパソコン不況のあおりを受け，現地パソコンメーカの大幅な減産が原因．

◆ ◆ 佐々木哲のニューヨーク ◆ ◆

N.Y. システム通信

何かがでてくる面白さ，Comdex

この原稿がのるころには，もう秋のComdexが始まっています．さて，今年は何が出てくるでしょうか？　もう新しいものはないという状況の中でも，何か新鮮なものがでてくる．これが米国のおもしろさです．

XGAの普及は来年以降？

SVGAを抑えてスタンダードとなれるか？IBMの次期グラフィック標準XGAに，最近動きがありました．現在，XGAはIBMだけがPS/2の上位MCA機種にのみサポートしています．今は，互換XGAがいつ出てくるかが普及の鍵です．英国の半導体メーカー，INMOSはIBMからXGAの独占生産・販売のライセンスを受けました．今後3年間，うち18か月は独占で製造するというものです．とりあえずは，MCA版のXGAチップセットはすぐに入手可能になるとか．問題のISAバス版は？　1992年の終わりまでにはとか．マイクロチャンネルとISAの構造の違いにより，ISA版のXGAではbus masteringはないでしょう．

とりあえず，マイクロチャンネル版のXGA互換カードは，この10月のComdexでもいくつかお目見えするようです．

一方，INMOS以外の会社でIBMとは関係なく，XGA互換チップの開発を進めるメーカーもあります．これらも，来年には登場することでしょう．VESAでも，ISAバスでのXGAの標準の検討もすすみつつあります．

このような状況なので，来年の終わりころまではXGAは広まりません．XGAと部分的な互換性をうたうカードは，いくつかでてくるでしょうが…….

S3を使ったグラフィックアダプタ

このような状況の中，アクセラレータ付きのVGAカードが次々に現れてきています．VGAとの互換性を保ちつつ，MS-Windowsなどを高速化する手ごろなカード群です．

今，このようなチップセットを販売あるいは開発しているのはATI，Chips&Technology，S3，Video Dynamics Inc.，Weiteckなどがあります．さらに，Windowsに特化したSVGAチップとしてもATI，Tseng，Headland，Western Degitalなどからでてくる模様です．

この中で唯一，製品として販売されているのは，ATIの8514/Ultra (Graphic/Ultra) でした．これに続いてS3 Inc.のチップセットを使ったボードの出荷が始まりました．

Orchid Technologyは，S3の86C911を使ったThe Fahrenheit 1280*[1]の出荷を開始しました．価格は，メモリ512KBのもので449ドルです．SVGAとしては，メモリアクセスが最速のTseng ET4000よりも，1ウェイトほど遅いだけの速度に加え，専用のWindowsやAutocadドライバなどを用いると，34020の数倍速い表示が可能．これはbitblts，直線描画など，数々の描画機能をハードウェアでサポートするGUIプロセッサならではのスピードです．高速なVideo RAMを装備，1MBのメモリを積めば，1280×1024の解像度での表示ができます．もちろんSierraのRAMDACを装備すれば16ビット，6万色の表示が可能です．

同様のディスプレイカードは次々にでてきます．Actix Systems Inc.もS3ベースのカードを発表しています．こちらは，512KBのメモリで399ドル．STB，Everexなども続きます．比較的高価だったATIのものの約半分近い値段で，このような高性能のカードが入手可能になってきました．

なぜ，S3が普及するか？　カードメーカーの開発を助けるチップメーカーの努力があります．各種ドライバ，BIOSなどプログラマーズキットも供給することにより，ハードウェアメーカーが少ない労力で完成品をまとめあげることができます．これは，今までのSVGAでも同様です．いいハードウェア，チップを作るだけではもう普及は望めません．

日本では？800×600のWindowsドライバ登場

米国では，このようにGUIが高解像度で高速に動く

ワンラインNEWS
メルコはPC-9800シリーズに対応した低価格のEMS RAMボード「EMJ-Rシリーズ」を発売した．1Mバイトから16Mバイトの容量で，ボード上にあとから増設もできる．

環境が整いつつあります．ところがわが日本は，せっかくDOS/V上のMS-WindowsでIBM AT互換機で日本語GUIを動かせる環境ができたのに，日本語をサポートしたドライバは日本IBM独特の寿命のきたDisplay Adapter/2に標準VGA用のものだけです．コプロセッサどころか，SVGA用のものさえない状況でした．これというのも，ただでさえ小さい日本市場を分断してのDOS/V，J-3100，AXの乱立が遠因でしょう．IBMは，すぐにXGAを搭載してくるでしょうが，DOS/V本領発揮のISAバス機種ではまだまだです．

この8月，このすぐとなりでコラムを連載している西川氏の会社，C.F.Computingがひとつのブレイクスルーを行いました．とりあえず800×600の解像度でのMS-WindowsドライバのOEM供給開始です．一般エンドユーザーへも，秋葉原方面のOEM先から販売されます．恵まれた米国の環境とは，けた違いの環境の中での努力に拍手です．

VGAを多色化するRAMDACの2社

現在，VGA多色化のためのRAMDACを発売してる2社のうちのひとつ，Edsunが行き詰まったようです．ほとんどの社員をレイオフし，すでに開発されたものなどAMDに譲り渡したようです．EdsunのRAMDACは，ダイナミックなパレットの制御を行うことにより，1MBのメモリのVGAで，74万色以上の表示を可能とし，さらにはグラフィックのスムージングまで行う画期的なものでした．

各種VGAカードに使用されただけではなく，EpsonやNorthgateの機種には，標準で採用されていました．ところが，このところのVGAの高解像度化，動作の高速化にともない，より高速の版のRAMDACの投入に遅れ，またこの特殊なモードをサポートするWindowsドライバの開発に150万ドルも使う結果となりました．

もう一方のSierraのRAMDACは，15ビットカラーを提供するものですが，16ビット版も登場が期待され順調なようです．

INTEL386，独占の終わり

INTELの386独占も終わりました．AMDの高速40MHz版を使った機種も値段で検討しています．また，まだまだ安定しないINTEL 386SLを後目に低消費電力のAMD 386SXを採用するノートブック型が増えてきました．その中でも，大手はToshiba Americaです．現在，20MHzまでの386SXをのせたノートブックT2200SXを発売している東芝ですが，次期モデルにAMD 386SXを採用することを発表しています．一方，日本でもマーケティングの混乱，日米の価格差の問題から遅れていたVGA搭載機種も登場間近のようです．

ドットプリンタでHP LaserJetを

今ではとても手軽になった，Hewlette PackardのLaserJetシリーズのレーザプリンタ．4ppmのLaserJet IIIpであれば1000ドル以下で手に入ります．これの互換プリンタ，あるいは一世代前のLaserJet IIpであれば500ドルでの入手も無理ではありません．このHP LaserJetをサポートしたソフトウェアは豊富にあり，これだけを対象にしたソフトもあります．

とはいえ，すでにEpsonなどのドットプリンタをもっているユーザー，HP社のDeskJetシリーズ，またこれのせっかくの日本版でありながら，Epson互換モードしかもっていないDeskJet 500Jのユーザーはどうすればいいのでしょう．

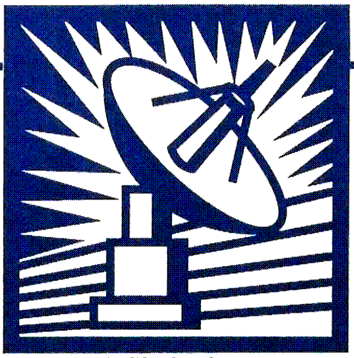
ソフトウェアだけでLaserJet IIをエミュレートするユーティリティがあります．Metro SoftwareのLaserTwin*2です．メモリに常駐してプリンタ出力を取り込み，いろいろなプリンタをLaserJetとしてエミュレートしてくれます．メモリ常駐サイズは26KBだけ．単なるプリンタ印字そのものだけでなく，HP独自のソフトフォント，ダウンロードフォントまでもサポートされています．なぜか，レーザプリンタでありながらLaserJetエミュレーションをもっていない米国CanonのLaser printerをサポートしたものもあります．

さらにはプリンタだけでなく，FAXボードをサポートするバージョンもあります．HPプリンタをサポートしたDOSアプリケーションなら，どれからでもFAXが送信可能．なかなかおもしろいところに目をつけるものです．これらの製品は，各所にOEMもされています．あなたの製品にHPエミュレーションが簡単に付加できるわけです．

*1 Fahrenheit 1280
Orchid Technology Inc., 45365 Northport Loop West, Fremont, CA 94538
phone (510) 683-0323
*2 All-In-One LaserTwin $149.00
PC/AccuFax $129
Metro Software, Inc., 1870 W. Prince Rd., Ste. 70, Tucson, AZ 85705
phone (602) 292-0313 fax (602) 292-1563

SYSTEM N Y NEWS

Illustration : NOBUKI FUKUHARA



西川和久の

希望的観測

いつまでも「漢字処理は特殊」では困る!!
PC上で健闘する日本語・2バイト対応ソフトを見る

最近，DOS/V騒ぎがちょっと一段落しているが，「フッ」と考えてみると，ソフトウェアだけでPC上に日本語環境を得る試みは，何もDOS/Vがはじめてだったわけでない．DOSレベルでのサポートは今回がはじめてなのであって，限定されたアプリケーションの内だけで日本語が使えるものは以前から存在した．今回は，DOS/Vに至るまでのいろいろなPC日本語環境の軌跡(？，といっても，まだ現在進行中であるが)を紹介する．

1 もっとも古くから海外で販売されている日本語ワープロソフト「EW＋」

もっとも古くから，海外で販売されている日本語ワープロとしてEW＋がある．VGAだけではなく，AT&T 640×400，Hercules，EGAなど，幅広いディスプレイタイプをサポートしているのが魅力である．原理としては，カナ漢字フォントをファイルとしてもっていて，起動時にメモリへ読み込み，ビットマップでグラフィック画面に描画する．DOS/Vと同じ作動原理であるが，DOS/Vのように，DOSレベルでサポートされているのではなく，ワープロ内だけのクローズされた世界でのみ漢字を扱うことができるのだ．

プリンタ出力も同じく，ビットマップで書き出している．即ち，24ドットフォントまでファイルでもっているのだ．日本語FEPは，E1相当が付属しており，EW＋内でだけ機能する．この手法は，その昔，パソコン上ではじめてワープロが出たころを思い出す．EW＋自体は，低価格ソフトで売り出している「アシスト・ワード」と同じものであり，ワープロの機能としては必要最小限といってよいだろう．

実際に使ってみると，80286クラスのCPUでも十分な速度が得られているし，DOSのメモリ空間が十分に空いていれば，EMS・XMSメモリも必要ない．日本語FEPを含め，使用感としては一世代前の気分であるが，通常の文書ならこれで十分であろう．DOS/Vの最低限のシステムリクエストを大幅に下回るため，カバーできる範囲はかなり広い．

2 通信・FAX機能まで備えたVGA日本語ワープロ「ユーカラArt」

この秋，発売になったばかりのワープロである．VGAのみのサポートであり，CPUパワーやメモリもそれなりに必要であるが，先のEW＋と比べると機能は明らかに上である．驚くことにワープロだけではなく，通信ソフトやコンピュータFAX(もちろんハードウェアは別)にまで対応している．DOS/Vで，ワープロ，通信ソフト，コンピュータFAXまで揃えると，けっこうな金額になるため，これ1つあれば，DOS/Vなどあえて必要ないという人も多いのではないだろうか!?　日本語FEPも，この中だけで使えるものが組み込まれており，全体としてのできばえもなかなかだ．面白いことに，DOS/Vの日本語環境下からそのまま起動できるため，パッ

ワンラインNEWS
日東電工はLCDの画面を見やすくする，視野拡大のための位相差補償ファイルを開発．素材の樹脂を改良し，液晶が重なる厚み方向に発生する着色現象を減している．

と見た感じでは，DOS/V対応アプリケーションと錯覚してしまう．

「どこかで見たことある画面だな!?」と思いながら触っていたのだが，思い出した!!　J-3100用アプリケーションで，「ビジコンポ」という統合型ソフトウェアのワープロと見た目そっくりだったのだ…….

3 韓・日・英語対応の超強力通信ソフト SuperSession（シェアウェア）

いままでは，国内で販売されているアプリケーションのPC日本語環境版だったが，今度のは凄い!　久々に興奮してしまったソフトウェアである．なんと，韓国で作られたPC，DOS/V，J-3100対応の2バイト圏用通信ソフトなのだ!!　驚くことに，半角，ハングル，日本語に対応していて，通信中，ダイナミックにヘルプメッセージを含むコードページを切り換えることができるのだ．PC上では，EW＋やユーカラArtと同様に，ハングル，日本語のフォントをもっていて，メモリへロードし，2バイト文字を描画する．VGAを拡張したSuper VGA，800×600の解像度もサポートしているため，半角で100×37という，広い画面上でも通信できる．しかも，DOS/VやJ-3100の日本語環境下では，それぞれのシステムからフォントを取り出してそのまま起動する．もちろん，日本語FEPまでも動いてしまう．これはショッキングな事件だ!!　まるで国産ソフトウェアのように振る舞ってしまうのだから…….

通信ソフトとしてのできばえも素晴らしく，見た目はモチーフ調，マウスオペレーション，スクリプトコンパイラ，ありとあらゆるプロトコル（Quick-VANにまで対応），ハイパーテキスト風ヘルプ，Telixに類似するコマンド形態など，初心者にはちょっと難しかもしれないが，通常の通信ソフトなど足元にも及ばない仕様に仕上がっている．ファイル全体で600KB近くあるが，NIFTYのFIBMPROにUploadされているので，動作環境がある人には絶対一度は見て欲しいと思う．世界感が変わるはずだ．

4 「何もそこまで……」といっていては「日本語環境」が世界においていかれる

国内のパソコンは，漢字ROMや漢字対応キーボードが当然のごとく搭載されているため，「何もそこまでしてPCで日本語環境を作らなくても……」という意見が大多数を占めるであろう．しかし，世界中で稼動しているPC，またその上で2バイト処理をしなければならない人々の数は膨大である．そして，SuperSessionのようなマルチリンガル，マルチ環境をサポートするソフトウェアの登場．いつまでも「漢字処理は特殊だから」といって逃げ回っていると，2バイト圏である隣国のソフトウェアが，ドーっと押し寄せてくると考えているのは私だけだろうか!?

Computer Laws

コンピュータの法律

プログラムの改造・その5

熊谷信太郎(弁護士)

■改造の程度による違い

前回に説明しましたように，プログラムを改造した場合，改造の程度によって，その後の法律関係が変わってきます．

(1)小規模の改造であり，従来のプログラムに対して新たな創造性がない場合には，プログラム著作権は従来の著作権者に帰属し，改造者にはなんの権利も発生しません．

(2)改造されたプログラムに，創造性が認められる程度の改造がなされ，かつ，改造後のプログラムがオリジナルのプログラムと実質的に類似している場合には，改造後のプログラムは2次的著作物となります．

この場合には，オリジナルのプログラムの著作権者の権利と，2次的著作物の著作権者の権利の双方が2次的著作物に及ぶことになります．

(3)プログラムの改造が極めて大規模で，オリジナルのプログラムと似ても似つかぬものであり，実質的類似性がない場合には，改造後のプログラムは2次的著作物ではなく，全く新しい著作物となり，改造後のプログラムは改造者だけが著作権を持つことになります．

■実質的類似性とは

このように，改造の程度によって，その後の法律関係が変わってくるのですが，その区別の際にポイントになるのが，「実質的類似性」という概念です．

「実質的類似性」とは，英語のSubstantial similarityの訳語ですが，著作権の及ぶ範囲はどこまでかを限定する機能がある概念です．つまり，著作権はオリジナルの著作物だけでなく，実質的類似性のある著作物にも及んでいるということになります．

これを逆にいえば，類似する著作物のすべてが著作権侵害になるわけではなく，実質的な類似性がある場合に限って著作権侵害となるということができるでしょう．

この「実質的類似性」の具体的内容は法律の条文には記載されていません．これは，どのような場合に実質的類似性があるといえるのか，あらかじめ法律を制定する際にすべてを予想して条文に個別的に記載しておくのは不可能だからです．結局，どのような場合に「実質的類似性」があるかというのは，具体的ケースごとに社会通念を基準にして両者が実質的に類似するか否かを判断していくしかありません．

■実質的類似性ー著作権侵害の基準

このような「実質的類似性」は，まず，類似する著作物が著作権侵害になるかどうかの際に基準となるという点に意味があります．たとえば，オリジナルのプログラムを参考にして，それとうりふたつの盗作プログラムを作ったという場合には，著作権侵害であることが明らかですが，もとの著作物をある程度は変えてあるという場合には，結局，どの程度似ているかが問題となり，その限界が「実質的類似性」があるか否かということになるわけです．

したがって，多少類似していても，「実質的類似性」がないという場合には，別個のプログラムであり，著作権侵害はないということになります．

■実質的類似性ー2次的著作物の基準

次に，「実質的類似性」は，2次的著作物といえるかどうかの基準としても意味をもっています．2次的著作物とは，前回説明したように，著作物に翻訳，編曲，変形，脚色，映画化するなどの翻案を加えて創作した著作物をいいます．

冒頭に説明したように，プログラムの改造が「実質的類似性」の範囲内ならば，改造プログラムは2次的著作物となり，原著作者と改造者の双方の著作権が及びます．

これに対し，プログラムの改造が大規模で「実質的類似性」の範囲を越え

ワンラインNEWS
シーエルシーは米EMCコーポレーションが開発したディスクアレイ「CLC7700」シリーズを発売．記憶容量は世界最大級，キャッシュメモリの採用で転送速度を速くした．

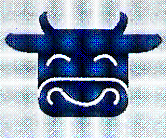
ている場合には，改造プログラムは2次的著作物とはならず，まったく新しいプログラムを製作したことになりますから，改造者の著作権だけが及ぶことになります．

アキバ短信

万世本店
アキバに笑う牛が帰ってきた

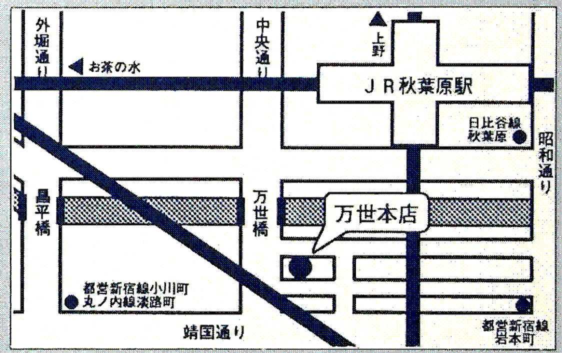
秋葉原の中央通りを駅から神田方面に進むと，万世橋の向こう側にそびえる10階建ての新しいビルが目にはいる．ビルのてっぺんでは赤い牛が笑っている．8月20日，ついにあの万世が秋葉原に本格的な復帰を果たしたのである．といっても，関東地方にお住まいの方以外はピンとこないだろう．万世は東京都内を中心に関東一円に店舗をもつ，ステーキ，焼き肉，すきやき，しゃぶしゃぶなどをはじめとする牛肉料理の店だ．「万世」という名前が示す通り，その本店は秋葉原の万世橋のたもとにある．「肉ビル」の通称で親しまれていた旧本店が取り壊されたのが数年前．以来，新ビルが完成するまでの間は裏通りの小さな仮店舗で営業していたが，このたび地下1階，地上10階の新築ビルが完成，万世新本店「MANSEI AKIHABARA」としてお目見えした．

新しい本店は，地下1階から10階にいたるまで，すべてのフロアが万世直営店で占められる．1階はコーヒーショップ「TRIM（トリム）」と売店．2階は中国料理の「桃李園」，3，4階はおなじみ洋食の「万世」，5階が焼肉の「万世牧場」，6階は調理室で7階はお座敷大広間と和・洋室を備えた「七福神」，8階は多目的ホール「TIARA（ティアラ）」，9階は和風個室の「味香華（みかげ）」，10階は鉄板ステーキの「千代田」，そして地下1階が「万世ラーメン」となっている．



笑う牛が目印

営業時間　開始時刻はフロアごとに異なる
ラストオーダー21:00　年中無休
TEL 03-3251-0291



ランチタイムには，フロアごとにランチメニューも用意されている．2階の「桃李園」では炒めもの，煮込み料理，揚げ物，辛い料理から1つを選んで，スープ，漬物，ライスとの組み合わせで1000円程度，3，4階の「万世」のランチメニューの中心は肉料理．和風ステーキ，和風ハンバーグなど（870円～2,700円）．7階はひさご弁当（2,700円）やしゃぶしゃぶ，すきやきのお気軽鍋（2,800円）．どこへ入ろうかと迷ってしまうほどだ．これで高いと思う向きには地下の「万世ラーメン」へ．おすすめはパーコーラーメン（740円）である．

夜ならぜひ3階以上のフロアを利用してみよう．万世本店は中央通りのどこからもよく見える．ということは，逆に本店からは秋葉原電気街が一望のもとに見渡せることになる．100万ドルとはいわないまでも，秋葉原の夜はネオンサインで華やかだ．

帰りには1階の売店へ寄ってみよう．ステーキ用牛肉や味付けハンバーグ，万世の生ラーメンなどもあるが，ここではやはり万かつサンド（460円）を買って帰りたい．笑う牛のクッキー（300円）は子供へのおみやげ向きか．

●プログラマの仕事場 | 8

Programmer's Working Space

マイクロプローズジャパン株式会社

「技術」だけでは「味」のあるゲームプログラムはできない

コンピュータゲームといえば，まだまだファミコン中心，ゲームの内容も小学生，中学生むけのものが多い中で，マイクロプローズジャパンは，各種の軍事シミュレーションゲームをはじめとする「大人のためのゲーム」を送り出している会社だ．IBM PC用ゲームの国産マシンへの移植も手がける同社をたずねて，ゲームプログラミングの魅力や，ゲームプログラマの実像について，代表取締役のコンラッド．T．小沢氏にお話をうかがった．

IBM PC用ゲームを国内のマシンに移植・販売する

マイクロプローズジャパンは，アメリカのソフトウェア会社マイクロプローズの日本法人である．といっても，あまりピンとこないだろう．しかし，「F-15 STRIKE EAGLE II」や「Red Storm Rising」の名前は，ご存知の読者も多いのではないだろうか．マイクロプローズジャパンの業務内容について，小沢氏は次のように語ってくれた．

「マイクロプローズは，アメリカのパソコンゲーム市場では，大きいほうから3番目の会社です．1位がブローダーバンド，2位がシエラ・オンライン，3位がマイクロプローズですが，シエラとマイクロプローズはあまり差がありません．僕らは，その業界3位のマイクロプローズのゲームを，日本でも少しでもお客様に知られるように，日本の機械に移植したり，開発・販売をやっています」．

マイクロプローズジャパンの仕事のひとつは，NECのPC-9800シリーズをはじめとする国産マシンへのゲームの移植である．驚いたことに，ひとつのゲームについて移植の担当者は1人だという．「98に移植したものは，5本あります．「パイレーツ」「ガンシップ」「サイレントサービス」「F-15」「レッドストームライジング」です．オリジナルのゲームを作るには，きちんとしたチームを組んでやりますが，移植の場合は，1人でやっています．アセンブラとCのソースがありますから，それを見て98用に打ちなおします」（小沢氏）．

ゲームソフトの制作体制はいまや映画なみ

マイクロプローズ社は，アメリカのゲーム業界では，すでにいくつものヒットを飛ばしている．新しいゲームソフトというのは，どんなふうに作られていくのだろうか．

「ゲームデザインからプログラミングまで1人でできる人は，もう化石のようなもので，ほとんどいません．昔は音楽からグラフィックスまで，全部自分でこなすという人もいましたが，いまはアメリカも日本もプロダクション的な作業になってきています．音楽，

ワンラインNEWS
日本シリコングラフィックスは3次元グラフィックWS「IRIS 4Dシリーズ」の低価格機種「RISC PC IRIS Indigo」を発売．同機は「ACE仕様」に互換性をもたせた．

プログラミング，グラフィックスなどにそれぞれ担当者がいて，シナリオデザイナーが監督のような形で別にいる．つまり，スペシャリストが担当する形になっています．

アメリカでの実際の開発についていうと，われわれの本社のほうでは，オリジナルのゲームを作っているわけですが，これには専業のゲームデザイナーがいます．『Railroad Tycoon』を作ったSid Maierというのはマイクロプローズの副社長ですが，彼はゲームデザイナー＋プログラマです．でも，アイデアのプログラマというべきでしょう．『これを画面に表現したい』という目的でプログラムを作っていますから，後でそのソースをほかのプログラマに渡しても読めるものではありません．『Railroad Tycoon』は，彼が『Covert Action』という秘密警察ゲームを作っているころ，休みをとってメリーランド州のRail Yardと呼ばれる昔の汽車が走っているあたりへ出かけたときに，子供のころの汽車へのあこがれを思いだして，その後わずか1週間で作り上げてしまったのだそうです．彼は1カ月ほどかけてゲームを作り上げては，ほかのプログラマに仕上げをまかせて，次のゲームに移ります」(小沢氏)．Sid Maier氏は，コモドール時代から活躍している有名人であるだけに，ゲームプログラマがオールマイティだった時代のスタイルをある程度残しているようだ．

現在，米国ではゲーム業界と映画が密接な関係にある，映画で有名なルーカスフィルムは，インディジョーンズシリーズなど，多くの人気ゲームを出しているし，映画界から人材を引き抜くゲーム会社もある．ゲームが映画製作的なスタイルになっているというより，両者が渾然一体となりつつあるといってもいいかもしれない．

「Sid Maier以外の人間はプログラマではなくて，ゲーム業界の人間です．つまりボードゲームのデザイナーをヘッドハントしたりしているわけです．そういう人ばかりを集めて，こんなゲームがあるとか，昔ああいったゲームを作ったことがあるけれど，あれをぜひコンピュータに載せたいなどという打ち合わせをして，ゲームを作っていくのです．まず，ゲームデザイナーがプロジェクトマネージャになって，手のあいているプログラマの中からそのプログラムに参加する人を募って，音楽担当－これは共通でいつも同じ人がやっています，次にグラフィックスの担当者であいている人がチームに加わります．そして，そのゲームデザイナーがチームリーダーとなって，いつまでに何を仕上げるか，発売日はいつごろ，テストプレーヤーはこのくらいほしいなどのスケジュールを組んでいきます」．

システムプログラマとゲームプログラマの違い

ゲームの発売日は，延び延びになりがちだ，とはよくいわれること．「メーカーの方からは，『ゲーム業界はこんなにいい加減なものなのか』と怒られたこともあるのですが，ゲームはほかのプログラムと違って，バグさえとればいいというものではありません．『味』がなければ成功しない．システム系のプログラムなどと大きく違うのはそこなのです．ですから，システムをやっているプログラマがどんなにできる人でも，ゲームを予定通りに作れるかといったら，それは疑問ですね」と，小

About Company

米国のパソコンゲーム業界で第3位のMICRO PROSE SOFTWARE Inc.の日本法人．代表取締役はコンラッド．T．小沢氏．MICRO PROSE社製品の日本での販売とPC-9800シリーズをはじめとする国内のパソコンへの移植，販売を行っている．

代表取締役　コンラッド．T．小沢氏

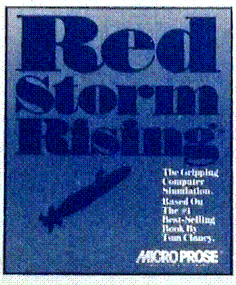
Products

「F-15 STRIKE EAGLE II」(IBM PC用9,900円，PC-9801シリーズ用10,800円)
全世界で100万本近く売れた「F-15 STRIKE EAGLE」の続編．プレーヤーが戦闘機F-15ストライク・イーグルのパイロットとなり，中近東，ベトナムなど実在の戦闘地域でMigやSAMと戦うフライトシミュレーションゲーム．

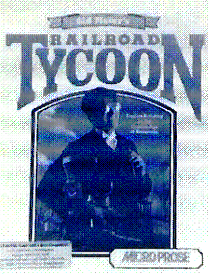
「Red Storm Rising」(IBM PC用9,800円，PC-9801用10,800円)
"レッドオクトーバーを追え"が映画化されたことでも知られるミリタリー小説のベストセラー作家，トム・クランシーの小説"レッドストームライジング"をコンピュータゲーム化した原子力潜水艦シミュレーションゲーム．プレーヤーは第3次世界大戦が勃発し，ソ連群が侵攻を開始したという設定の下，アメリカ軍の最新式原子力潜水艦

ワンラインNEWS
富士通は記憶容量が80Mバイトと世界最大の2.5インチHDDの開発にメドをつけ，今秋にも市場に投入する．データショウに出展後サンプル出荷を開始する予定．

の船長となって，ソ連軍の北洋艦隊と戦う．ゲーム中では現実に存在する兵器をそのまま再現しているため，リアルな現代戦を体験できる．



「RAILROAD TYCOON」(IBM PC用10,900円，Macintosh用12,900円)
鉄道ゲームとマネーゲームの要素がミックスされた，鉄道会社経営ゲーム．プレーヤーは鉄道経営者となって，自ら鉄道の開拓場所を選択し，わずかな資本金をもとに線路を引き，列車を走らせる．ゲームの中に存在する同じ目的を持ったライバル会社に負けないよう，新しい地域のどのルートに線路を引くのか，どんな種類の列車を走らせるのか，設備をどんなふうに改良するのか，株式購入のバランスはどうするのかなどを考慮しながら，最良の鉄道経営者をめざす．歴史に残る有名な列車が登場したり，汽笛や蒸気の音などさまざまな効果音も楽しめる．PC-9801用も近日発売の予定．



沢氏は話してくれた．

「それに，ゲームプログラマとシステム開発のプログラマというのは，同じ技術レベルをもっていないと成り立たないのですが，『すごい』といわれるゲームプログラマをシステムハウスにつっこんで，逆に一番優秀なシステムプログラマをゲームハウスに入れたら全然仕事にはなりません．やはり真水の魚と塩水の魚という感じではないでしょうか．」

小沢氏によると，米国とはソフトの販売本数が2桁違うという，狭い日本の業界ではあるが，それにもかかわらず人材不足は悩みの種である．「プログラマを募集すると，自称プログラマという人が集まってきます．『Z80のアセンブラをやっていた』という人もきましたが，結局彼は全部のレジスタが正常にデータを運んでいるかどうかというテストプログラムを作ったことがあるだけでした．これではゲームは作れません．システムプログラマというのは，全部を自分で作り上げることができない．これは，システムプログラマとゲームプログラマの大きな違いですね．」プロダクション的になり，分業が進んでいるとはいっても，ここ一番で「味」を求められるゲームプログラマはプログラミングの技術だけではつとまらないようだ．

ひとつ目玉があれば売れる時代は終わったが……

70年代後期のマイコンブームのころからコンピュータに触れてきたという小沢氏．それだけに，時代の移り変わりを実感しているようだ．

「昔，ソードM5というパソコンがありましたが，これのBASICをハンドアセンブルでたったの1週間で作り上げたという人を知っていますが，この人などは本当にすごいと思いました．それに，PC-88の高速画面描画ルーチンをもっているという人もいました．1画面を3分くらいで描きなおすのです．いまでは問題にならないスピードですが，当時はそれでも，ものすごく速いといわれていました．」ゲームにしても，描画速度の速さ，絵のきれいさなど，ひとつでも目玉があれば売れた時代だったという．

「『平安京エイリアン』という一世を風靡したゲームは，京都大学の学生が作ったということで，誰もがゲームデザイナーになれるのではないかと思いましたよね．昔は，そういう意味でもオールラウンドなプログラマがはやりだったのです．いまの日本には，何でもできるオールラウンドのプログラマはいなくなってしまいました．ゲーム音楽専門のプログラマやグラフィックス専門のプログラマなどに分担が分かれてきています．その意味では，システムプログラマもゲームプログラマになれないことはありません．」

時代は激しく移り変わり，人材不足という問題がある中でも，小沢氏はコンピュータゲームの将来は明るいと考えている．「コンピュータが普及しても，家庭ではワープロを少し使うくらいで，表計算ソフトで家計簿をつける必要はないし，何をやるんでしょう？」というわけである．

なお，同社では随時プログラマを募集中とのことである．アイデア勝負，そして味のあるゲームプログラミングの世界へ華麗なる(?)転身をはかりたい人は，チャレンジしてみてはどうだろうか．

ワンラインNEWS
松下通信工業は45MB HDDを内蔵した業務用ワープロ「パナワード5800D」を発売．同機種は5000シリーズの最上位機種で，従来機の5.5倍の文書容量をもつ．

information・page

イベント

■10月22日(火)~10月25日(金)
データショウ'91
"人にやさしい情報環境の想像"をテーマに164社がハードウェア・ソフトウェアを展示.
会場:東京国際見本市会場(東京・晴海)/入場料:無料
問い合わせ先:データショウ事務局 TEL 03-3433-4547

■10月26日(土)・11月11日(月)~11月15日(金)
NICOGRAPH '91
コンピュータグラフィックスに関するハードウェア・ソフトウェアの展示のほか
セミナーも行われる. 10月26日には記念シンポジウム, 11月11日には特別講演が行われる.
会場:池袋サンシャインシティ(東京)他/入場料:1,000円
問い合わせ先:(社)日本コンピュータグラフィックス協会 TEL 03-3233-3475

■11月28日(木)~11月30日(土)
NECパソコンフェア'91
日本電気と周辺機器・ソフトウェアメーカーによる展示会.
会場:日本コンベンションセンター(幕張メッセ)
問い合わせ先:日本電気(株) TEL 03-3454-1111

海外イベント

■10月21日(月)~10月25日(金)
COMDEX/Fall '91
約1800社が出展する最大規模のコンピュータ関連製品の展示会.
会場:Sands Expo and Convention Center(米国・ネバダ州ラスベガス)
問い合わせ先:ザ・インターフェイス・グループ日本支社 TEL 03-3237-8759

■11月13日(水)~11月15日(金)
NORTHEAST COMPUTER SHOW
イーストウェストを中心としたハードウェア・ソフトウェア, 関連製品の展示会.
会場:World Trade Center(米国・マサチューセッツ州ボストン)
問い合わせ先:ザ・インターフェイス・グループ日本支社 TEL 03-3237-8759

セミナー

■10月29日(火)~10月30日(水)
第3回X Window System Version 11 Release 5 テクニカル・セッション
上級エンジニアを対象にX Window Systemの最新バージョンに関する
最新技術情報を具体的な例とともに解説.
会場:子供の城(東京・渋谷区)
参加費:58,000円(2日間・昼食, テキスト代含)
問い合わせ先:(株)創夢 TEL 03-5453-1251

■11月26日(火)~11月28日(木)
オブジェクト指向設計とC++セミナー
オブジェクト指向技術によるC++プログラミングを
体験しながら学習する.
会場:虎の門パストラル(東京)
受講料:190,000円(3名以上, 2講座受講者に割引あ…
問い合わせ先:ソフトウェアテクノロジー インターナショナル TEL 03-5…

■10月31日(木)
386ネイティブCコンパイラ実践プログラミングセミナー
386/486をフルに使いきるプロテクトモードプログラミングを
WATCOM C/386を例に解説.
会場:東京YMCAホテル(東京・千代田区)
受講料:20,600円(消費税・資料・昼食代含)
問い合わせ先:(株)ライフボート
「386ネイティブCコンパイラ実践プログラミングセミナー」係
TEL 03-3293-4714

■11月21日(木)
活用されるためのCASE 構造化手法実践セミナーin大阪
構造化分析・設計の全体図, リアルタイム構造化手法などを
Kanga-Toolを用いて解説する.
会場:大阪駅前第4ビル20F(大阪市・北区)
受講料:30,900円(消費税・資料・昼食代含)
問い合わせ先:(株)ライフボート
「活用されるためのCASE—構造化手法実践セミナーin大阪」係
TEL 03-3293-4714

■11月22日(金)
リアルタイム制御プログラミングセミナーin大阪
リアルタイムシステムの概念, 構築方法, MS-DOSシステムからリアルタイムシステム
への拡張方法などをLattice C/RTシリーズを例に解説する.
会場:大阪駅前第4ビル20F(大阪市・北区)
受講料:30,900円(消費税・資料・昼食代含)
問い合わせ先:(株)ライフボート
「リアルタイム制御プログラミングセミナーin大阪」係
TEL 03-3293-3887

Microsoft University

■10月28日(月)~11月1日(金), 11月11日(月)~11月15日(金)
Microsoft Windows Programming Environment
Windowsアプリケーション開発のための入門コース.
会場:特別会場/受講料:150,000円(5日間)

■10月28日(月)~11月1日(金), 11月25日(月)~11月29日(金)
MS OS/2 Presentation Manager Applications Programming
OS/2 PMアプリケーション開発のための上級コース.
会場:特別会場/受講料:150,000円(5日間)

■11月5日(火)~11月8日(金)
MS OS/2 Programming Environment
OS/2アプリケーション開発のための基本コース.
会場:特別会場/受講料:120,000円(4日間)

■11月18日(月)~11月22日(金)
MS OS/2 Applications Programming
MS OS/2アプリケーション開発のための上級コース.
会場:特別会場/受講料:150,000円(5日間)

■11月11日(月)~11月12日(火)
C言語入門
QuickCを使ったC言語の入門コース.
会場:特別会場/受講料:60,000円(2日間)

■11月6日(水)~11月8日(金)
C言語応用
MS-C 6.0を使ってC言語の高度な技術を学ぶ実習中心のコース.
会場:特別会場/受講料:90,000円(3日間)

■11月18日(月)~11月22日(金), 11月25日(月)~11月29日(金)
Microsoft Windows Applications Programming
Windowsアプリケーション開発のための上級コース.
会場:特別会場/受講料:180,000円(5日間)

■11月19日(火)~11月22日(金)
Microsoft LAN Manager Programming Environment
LANマネージャアプリケーション開発のための入門コース.
会場:特別会場/受講料:150,000円(4日間)

■11月25日(月)~11月29日(金)
Microsoft LAN Manager Applications Programming
LANマネージャアプリケーション開発のための上級コース.
会場:特別会場/受講料:180,000円(5日間)

Microsoft Universityは, OS/2, Windowsアプリケーション開発者のために
定期的に開催されているセミナーです.
問い合わせ先:富士ソフトウェア(株)MU教育センター TEL 03-3452-7…

NEWS & NEW PRODUCTS

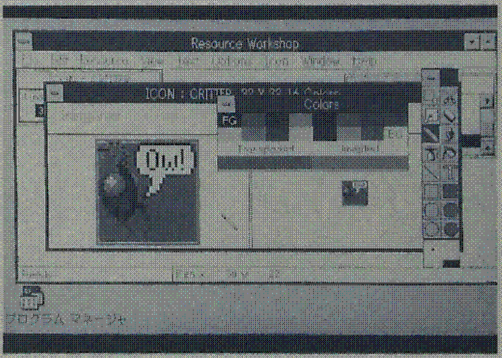
ボーランドジャパン，98用Turbo Pascal for Windowsを発売

（株）ボーランドジャパン（TEL 03-3486-1400）は，SDK（Software Development Kit）を使わずにWindowsアプリケーションの開発が行える「Turbo Pascal for Windows 1.0」を10月下旬より発売する．Turbo Pascal for Windowsは，アプリケーションの骨格となるクラスライブラリObjectWindowsが付属しており，このクラスライブラリからウィンドウメニュー，ダイアログ，コントロールなどを継承すれば，最初からコードを記述する必要がない．ソースコードの編集，コンパイル，実行，デバッグなどは，すべて統合環境を用いてWindows上で行うことができる．ダイアログ，メニュー，ビットマップ，カーソルなどのWindowsリソースを視覚的に作成・編集ができるリソースワークショップ，マイクロソフト社からライセンスを受けているリソースコンパイラ，ヘルプコンパイラ，WINDOWS.Hなども付属している．

また，同社では，PC-9801版の「Turbo C++ 2nd Edition」を20,000円（旧価格39,800円）に，「Turbo Pascal 6.0」を20,000円（旧価格29,800円）に，「Turbo Pascal 6.0 Professional」を39,800円（旧価格59,800円）にそれぞれ価格改訂するほか，学生個人の使用に限定した「Turbo C++ 2nd Edition」，「Turbo Pascal for Windows」（各10,000円）のスチューデントパックを発売し，すでに発売中のほかの言語製品のスチューデントパックと1セットのマニュアルと10セットのライセンスを組み合わせたラボパックについても価格の見直しを行った．

■おもな仕様

動作環境：MS-DOS 3.01以上，MS-Windows 3.0以上（リアルモードではインストーラ・統合環境・デバッガが動作しない），メモリ2.5MB以上，ハードディスク9MB以上／適応機種：PC-9801，PC-98，PC-H98シリーズ，PC-286/386シリーズ／価格：29,800円



サン・ソフト社，分散環境用システムソフトウェアSOLARISを発表

米国サン・ソフト社は，SPARCシステムと80386，80486を採用したシステム上で，UNIXをベースにした分散コンピューティング環境を実現する総合システムソフトウェア「SOLARIS」を発表した．「SOLARIS」は，UNIXオペレーティングシステム「SunOS」，ネットワーク体系「ONC（オープン・ネットワーク・コンピューティング」，ウィンドウ環境「OpenWindows」によって構成され，分散コンピューティング環境を前提としたトータルなシステム環境を実現している．米国内で即日出荷予定のSPARC用「SOLARIS 1.0」は，SunOS 4.1.1，OpenWindows2.0，ONCで構成されており，1992年前半出荷予定のSPARCおよび80386/486用「SOLARIS 2.0」は，UNIXシステムVリリース4をベースに開発されるSunOSの次期バージョンSunOS 5.0，機能拡張されたONC，OpenWindowsの次期バージョンOpenWindows 3.0で構成される予定．また，ソフトウェア開発者に対しては，分散アプリケーションの開発に適したオブジェクト指向型ツールを備えた強力な開発環境を提供する．国内では日本サン・マイクロシステムズ（TEL 03-3221-7021）を通して日本語化したSPARC版SOLARISを提供する予定．

東芝，J-3100対応のネットワーク用OSを発売

（株）東芝（情報処理・制御システム企画室：TEL 03-3457-8145）は，同社のJ-3100シリーズに対

ワンラインNEWS
メモレックステレックスは日本IBMの「PS/55シリーズ」に対応した日本語と英語の両方を出力できるバイリンガルドットプリンタ「MEMOREX TELEX 2165」を発売した．

NEWS & NEW PRODUCTS

応したパソコンネットワーク用OS,「NetWare386 v3.1J」,「日本語MS OS/2 LANマネージャ V2.0」,「NET/GC2 LANマネージャ」の３種を発売した．

「NetWare386 v3.1J」は,米国ノベル社の「NetWare」の日本語版で,サーバのメインメモリの一部をキャッシュメモリとして利用することにより,高速なデータのやりとりが行え,データの索引をコード化するなど,効率的なサーバ処理が可能.「日本語MS OS/2 LANマネージャ V2.0」は,メーカー18社で構成される日本語MS LANマネージャ共通規約に準拠し,他社のパソコンを端末として利用できるマルチベンダー対応を実現している.「NET/GC2 LANマネージャ」は,同社独自の基本ソフト「NET/GC」の機能を「日本語MS OS/2 LANマネージャ」に付加したもので,電子メール,電子掲示板,会議室管理ソフトなどの機能をもつ.

エプソン，ハイコストパフォーマンスの16ビットデスクトップを発売

セイコーエプソン(株)，エプソン販売(株)（エプソンPCインフォメーションセンター：TEL 03-3377-3531）は，CPUに80286（16MHz），５インチFDDを２基搭載し，20万円前半の価格を実現した．ハイコストパフォーマンスの普及型16ビットデスクトップパソコン「エプソン PC-286VJ」を発売した．サウンド機能を内蔵し，着脱可能なカートリッジ型高速ハードディスクユニット(20MB，40MB，80MB)，SCSI対応ハードディスクユニット(100MB，200MB)も使用可能．

■PC-286VJ-5/5B

プロセッサ：80286（16MHz）／メモリ：640KB（最大14.6MBまで拡張可能)／外部記憶装置：５インチFDD×２(VJ-5)，５インチFDD×２・40MB HDD×１（VJ-5B）／価格：228,000円（VJ-5)，358,000円（VJ-5B）

富士通・PFU，RISCを採用したUNIXコンピュータ7機種を発売

富士通(株)（システム第二販売推進部：TEL 03-3216-8004）と（株）PFU（東京本社営業支援部：03-3438-0311)は，「UNIXSystem V リリース４」とRISCアーキテクチャのSPARCを搭載したUNIXコンピュータ「DS/90 7000シリーズ」全７機種11モデルを発売した．このシリーズは，SPARC 25MHz/33MHz/40MHzの採用，2CPUマルチプロセッサのサポートにより，18MIPS～58MIPSの性能を実現している．サーバタイプのDS serverは，150MB 1/4インチカートリッジを標準装備し(7530を除く)，小型デスクトップ，デスクサイド，ラックマウントの３タイプ，４機種，ワークステーションタイプのDS stationは，ラップトップタイプとデスクトップタイプ，３機種がある．

■DS server7530/7630/7730/7732/7830/7832

プロセッサ：SPARC/40MHz（7732/7832は2CPU）／メモリ：16MB～最大512MB／外部記憶装置：3.5インチFDD×１・520MB～671MB HDD×１（最大29.5GB）／価格：2,400,000円～

■DS station 7110/7120/7220/7230

プロセッサ：SPARC/25MHz，33MHz，44MHz／メモリ：16MB～最大96MB／外部記憶装置：3.5インチFDD×１・180MB～520MBHDD×１(最大3.6GB）／価格：1,400,000円～

NEWS & NEW PRODUCTS

日本サン・マイクロシステムズ，SPARCstation初のラップトップモデルを発売

日本サン・マイクロシステムズ(株)（TEL 03-3221-7021）は，プロセッサにSPARC(25MHz)を搭載し，17.5MIPS，1.8Mflopsの高速性能を実現したSPARCstation初のラップトップモデル「SPARCstationLT」2機種を発売した．11.5インチELディスプレイを採用したELタイプと，12インチバックライト付きモノクロ液晶ディスプレイを採用したLCDタイプがある．メモリは標準で8MB，最大40MBまで拡張可能で，329MBハードディスクを標準装備している．オペレーティングシステムにはSPARCstationシリーズと共通の「SunOS」を採用し，ウィンドウ環境には「日本語OpenWindows」を標準装備している．また，小型ながら，拡張インターフェースSBusスロット，イーサネットインターフェース，2つのシリアルポート，プリンタポートを備える．価格は1,780,000円(標準システム)．

日本エヌ・シー・アール，68040搭載のUNIXマシンを発売

日本エヌ・シー・アール(株)（TEL 03-3582-6111）は，同社のビジネスUNIXコンピュータ「NCR TOWERファミリー」の追加モデルとして，キャッシュメモリや浮動小数点演算プロセッサなどを内蔵した32ビットマイクロプロセッサMC68040（25MHz）を搭載した「NCR TOWER32/750」を発売した．従来機のTOWER32/700にくらべて約2.5倍のパフォーマンスを実現し，プロセッサボードにSCSIを内蔵することにより，周辺機器のコントロールに幅をもたせ，マルチタスク処理を容易にしている．価格は，8MBのメモリ，150MBカートリッジテープ，380MBハードディスク，OS込みの基本構成で10,450,000円．

三菱電機，68040搭載の低価格CISCマシンを発売

三菱電機（株）（情報処理機器部：TEL 03-3218-3226）は，プロセッサにMC68040（25MHz）を搭載した三菱エンジニアリングワークステーション「MELCOM MEシリーズME520」を発売した．処理速度約20MIPS，4M DRAMのASIC LSIの搭載により高さ69mmの薄型筐体を実現した．200MBのハードディスクを搭載し，メモリは標準で8MB，最大64MBまで増設可能．UNIX System Vに4.3BSDの機能を取り入れたME/UX-II，X-Window X11R4，OSF/Motif1.1などを標準装備している．価格は，2,543,000円．

沖電気工業，LED光源，PostScript搭載のページプリンタを発売

沖電気工業（株）（TEL 03-3501-3111）は，LED（発光ダイオード）を光源に用いた電子写真方式で，Adobe Systems社のページ記述言語PostScriptを搭載した日本語ページプリンタ「MICROLINE801PS PLUS」を発売した．5書体の日本語アウトラインフォント，35書体の欧文フォントを標準装備し，これらを100MBの内蔵ハードディスクに格納している．B4版400dpiの高精細印刷が可能で，マッキントッシュ用，Windows 3.0用プリンタドライバを標準添付，SCSIで外部フォントハードディスクと接続が可能．価格は，1,148,000円．

マイクロソフト，Windows 3.0対応のExcel，Wordを発表

マイクロソフト(株)（インフォメーションセンター：TEL 03-3363-1201）は，Windows 3.0対応

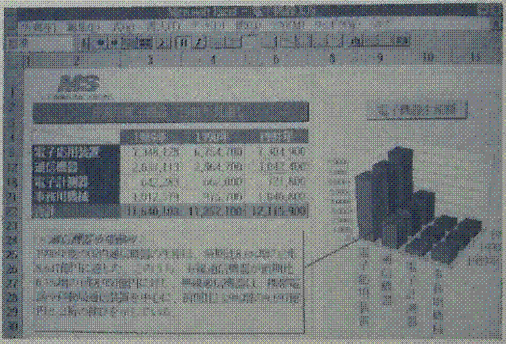
ワンラインNEWS
シャープはひとつのチップ上に16ビットMPU，ROM，キーボードインターフェースなどを一体化したシステムオンチップを開発した．

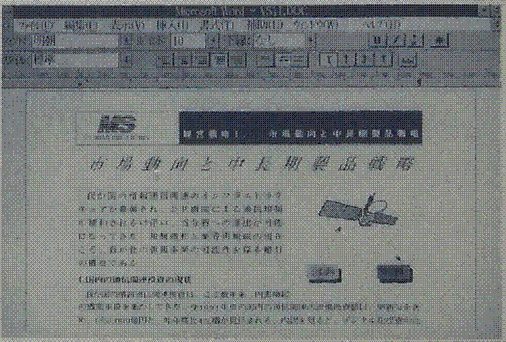
NEWS & NEW PRODUCTS

の統合型表計算プログラム「Microsoft Excel for Windows Version3.1」（写真上，以下Excel 3.1）を10月18日より，日本語ワードプロセッサ「Microsoft Word for Windows Version1.2」（写真下，以下Word 1.21）を11月下旬より発売する．

「Excel 3.1」は，表計算，ビジネスグラフ，データベースの3つのツールを有機的に結合させた統合型表計算プログラムで，頻繁に使う操作をボタン化したツールバーやオートSUMボタンの採用，広大なワークシートから必要な項目だけを抽出できるアウトライン機能，複数のワークシートをくしざし演算する統合機能，データの最適化シミュレーションを手軽に実現するソルバー機能，3次元グラフの作成など，さまざまな機能強化がなされた．価格は98,000円（バージョンアップ価格：25,000円）．

「Word 1.2」は，同社が提供するはじめての日本語ワードプロセッサで，下書き感覚で入力できる高速モード，書類全体の文章構成を考えるときに便利なアウトラインモードを装備し，文書作成プロセスに柔軟に対応することができる．文字の書式を設定するコマンドを集めたリボン，段落の書式を設定するコマンドを集めたルーラー，設定された書式を一括して登録できるスタイル機能など，DTPに迫る機能をもつ．価格は58,000円．





日本IBM，XGAの推進を発表．OADGなど対象に．

日本アイ・ビー・エム（株）（TEL 03-3586-1111）は，パソコンソフトウェアの共通利用を目指して同社が推進している「PCオープン・アーキテクチャー」の画面制御論理として，現行のVGA（Video Graphics Array）に加えて，より高度なグラフィック機能をもつXGA（eXtended Graphics Array）も推進することを発表した．

XGAは，VGAとの上位互換性を保ちながら機能強化した次世代の制御論理で，1024×768ドットの解像度，26万色中256色を同時表示することが可能で，処理速度も高速化している．

アップルコンピュータ，アドビとフォント技術の協力体制を強化

米国アップルコンピュータ社とアドビシステムズ社は，アドビシステムズ社のType1フォント技術を将来のSystem7に含むことに合意，その覚書を取り交わしたことを発表した．これは，System 7の新機能のひとつであるアップル社のアウトラインフォント技術「TrueType」と，アドビシステムズ社の「Type 1」の両方のフォントを使用できるようにするという指針を示したもので，この合意により，両社はMacintoshのシステムソフトウェア内で，Type1のラスタライザとType1のフォントをサポートするために必要なソフトウェアを開発する意向であることを確認した．また，アップル社はMacintoshのシステムおよびプリンタの購入者に，Adobe Type Managerのラスタライザと数種類のType 1フォントを暫定的に提供する意向であり，1991年秋から実施する予定である．

Programmer's Program

80486までのCPUに対応したアセンブラソースジェネレータ

ZZ386

ZZ386は，8086から80486までのCPUに対応した，アセンブラソースジェネレータである．ファイルタイプは，拡張子にCOMやEXEをもつ実行形式ファイルだけでなく，デバイスドライバ・オブジェクトファイル，拡張子にEXPをもつDOSエクステンダ用の実行形式ファイルにも対応している．

しかし，ZZ386の最大の特徴は，スクリーンエディタの操作感覚で正確なソースコードを得ることができるという点だろう．

インストール

パッケージは，約60ページのマニュアルが1冊，5インチと3.5インチのフロッピーディスクそれぞれ1枚から構成されている．また，ZZ386のパッケージには，PC-9800シリーズ版とPS/55シリーズ(DOS/V)版の両方が含まれている．ただし，DOS/V版は英語モードでは使えないので注意が必要である．

インストールは，マスターディスクに入っているINSTALLプログラムを起動して，マスターディスクの入ったドライブとインストール先のドライブを入力すればいい．いたって簡単である．

ZZ386を起動するには，ターゲットとなるプログラムをカレントディレクトリにおき，MS-DOSのコマンドラインから，

```
A> ZZ386
```

と入力する．まず，プロジェクトファイルの入力を求めてくるので，ファイル名を指定する．以後このファイル名が，生成されるソースやクロスリファレンスのファイル名として使われる(拡張子は.ASMなどになる)．もちろん，プロジェクトファイルを引数に指定して起動することも可能である．

プロジェクトファイル名を設定すると，プロジェクトを設定するサブメニューが開く．ここで，ターゲットのファイル名やファイルのタイプ，ワークエリアなどを設定する．設定終了を選択するとターゲットとなるファイルが読み込まれ，スクリーンに逆アセンブルリストが表示される．この時点では，すべてを命令としてスタートアドレスから単純に逆アセンブルしたソースコードである．

まずコードの自動解析

ソースコードの解析は，まず自動解析によって行う．この操作は「メイク」と呼ばれ，[f・10] または [M] キーを選択して行う．このコマンドによって，スタートアドレスから命令を解析し，ラベルを付加し，命令とデータの領域を分離していく．ターゲットが単純なコードであれば，この作業だけでソースの生成は完了である．

生成されたソースコードをファイルへ出力するには，[f・9] または [U] キーを選択する．アセンブル可能なファイルを出力するには，[f・9] または [U] キーを選択したのち，まず [O] (オプション)を選択し，出力ファイル形式を設定する必要がある．

ソースコードの修正

ターゲットが小さなプログラムなら，ソースジェネレータで正確なソースコードを得ることができる．しかし，最近のコンパイラは高度な最適化機能などをもち，命令やデータを複雑に組み合わせたコードを生成するので，簡単に正確なソースコードを得ることはできない．通常なら，生成されたソースコードをエディタで読み込み，クロスリファレンスを参照しながら修正する作業が待ちかまえている．

しかし，ZZ386ではスクリーン上でデータとコードの指定変更，ラベル名の変更・付加などをエディタ感覚で簡単に行うことができる．変更した部分と関係のあるほかの部分も，自動的に修正されるのはいうまでもない．さらに，絶対アドレス・各セグメントのオフセットとしたオペランド値・ラベル参照値などへのジャンプ機能や特定インストラクション・アドレスや数値・

ワンラインNEWS
大塚商会はWS用OSであるUNIXの学習ソフト「UNIX SELF MASTER」を発売した。ソニーのNEWS上で動作し，MS-DOSとコマンド比較をしながら学習を進められる。

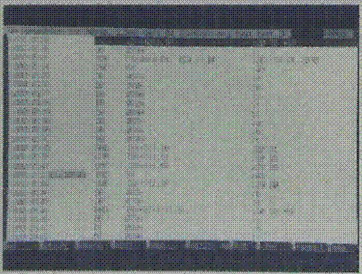
文字列などの検索もできる．ヒストリバッファをもっているので，ジャンプ前の位置に戻ることも可能だ．

通常のソースジェネレータで生成されたソースコードを，エディタで修正する作業に比較してみれば，ZZ386の操作効率のよさがわかるだろう．

単なるソースジェネレータではない

もちろん，ZZ386は全角文字が各所に入ったターゲットにも対応している．自動解析によって全角文字列を命令や数値配列として解釈し，ソースコードの修正に手間取るといったことはありえない．あたりまえの機能のように思えるが，対応していないジェネレータも多いのである．

ほかにもソースコードの部分解析，クロスリファレンスリスト，ダンプリストと，ターゲットの解析に必要な機能がZZ386には揃っている．ZZ386は，単なるソースジェネレータではなく，ソースコードを得るための「統合環境」といったものなのである．



■ZZ386

販売：（株）マイクロ・オーグ（TEL 03-3375-3306）／価格:39,000円（消費税含まず）／適応機種:日本電気PC-9801シリーズ，エプソンPC-286/386シリーズ，日本IBM PS/55シリーズ(DOS/V)／動作環境:MS-DOS Ver.3.1以降，ハードディスクでの使用を推奨／必要メモリ:640Kバイト，EMSメモリ対応／供給メディア:5インチ2HD，3.5インチ2DD

注）評価版のディスク内容構成は，現行の製品版と異なる可能性がある．評価版ではクロスリファレンスが出力できなかった．

リスト1 「グラフィック画面の出力を止めるプログラム」(GOFF.ASM)

```
;====================================
;==  GOFF                          ==
;==                                ==
;==  Copyright 1989(C) by Yoshi    ==
;====================================

seg_a           segment byte public
                assume  cs:seg_a, ds:seg_a

                org     100h

main            proc    far

start:
                mov     dx,offset msg
                mov     ah,09h                  ; DOS Services  ah=function 09h
                int     21h                     ;  display string
                mov     al,0ch
                out     0a2h,al
                mov     ah,4Ch                  ; DOS Services  ah=function 4Ch
                int     21h                     ;  terminate with al=return code

main            endp

;========================================
;           Messages area
;========================================

public      msg

msg:    db  'ｸﾞﾗﾌｨｯｸ画面表示停止 GOFF.COM ver.1.0'
        db  ' Copyright 1989(C) by Yoshi',0ah,0dh,'$'

seg_a           ends

                end     start
```

リスト2 「ZZ386で自動解析したソースコード」(GOFF.ASM)

```
        .8086
        .8086
        INCLUDE GOFF.EQU

START_SEG Segment Para Public
Assume CS:START_SEG , DS:START_SEG , SS:START_SEG
Org 00100H

GOFF  proc  Far
        ;プログラムの実行開始点
PRG_START:      MOV     DX,Offset ES_1010F      ; BA 010F 3.
                MOV     AH,009H                 ; B4 09 ｨ.
        ;文字列の出力 AH=09 文字列(DS:DX)-->
                INT     021H                    ; CD 21 ﾍ!
                MOV     AL,00CH                 ; B0 0C ｰ.
                OUT     0A2H,AL                 ; E6 A2 鋤
                MOV     AH,04CH                 ; B4 4C ｴL
        ;子プロセスの終了 AH=4C AL=Ret Code.
                INT     021H                    ; CD 21 ﾍ!
GOFF  endp

ES_1010F        DB      'ｸﾞﾗﾌｨｯｸ画面表示停止 GOFF.COM ve'
                DB      'r.1.0 Copyright 1989(C) by Yoshi',00AH,00DH
                DB      024H                    ; $

START_SEG Ends
END     PRG_START

^L
```

オンラインソフト採集帳 11
バッチファイルからCOMファイルを生成する BATCP

「プログラムを書くことができますか」と聞かれて「できない」と答える人でも，「バッチファイルを書くことはできますか」と聞かれれば「バッチファイルなら」と答える人がいるだろう．そんなあなたに，「プログラムが書ける」といえるようにしてくれるツール，それがBATCP（BATCH FILE Compiler）だ．

バッチファイルをコンパイル（?）

BATCPは，バッチファイルをコンパイルするツールである（正確にいえばCコンパイラのようにコードを生成するのではないが，便宜上ここではコンパイルと呼ぶ）．バッチファイルをコンパイルすることで，バッチファイルのいくつかの制約を回避することができるようになる．

- コマンド入力後，そのファイルの入っているドライブからフロッピーディスクを交換できるようになる．
- スクリプトをいちいちディスクから読み出さないので動作が速くなる．
- COMファイルなので，自由にバッチファイルが呼び出せる．
- コマンドが拡張される．

などのメリットがドキュメントにも書かれている．この中で，とくに恩恵にあずかれるのは，「わたしはプログラムができるんだぞ」と自慢できることではないかと思う．

コンパイル方法

コンパイルしたいバッチファイルのあるディレクトリに移動し，MS-DOSのコマンドラインから

A> BATCP ［ファイル名］

と入力する．これでCOMファイルができあがる．なお引数に指定するファイル名には拡張子を付ける必要はない．

BATCPで生成したCOMファイルの実行時には，環境変数「COMSPEC=」の設定，および設定されたコマンドプロセッサ（通常はCOMMAND.COM）が必要である．

また，常駐プログラムを実行するためのバッチファイルをコンパイルして実行しても，メモリを浪費するだけなので避けたほうがいい．

BATCPには，次のようなオプションが用意されている．

［書式］
A>BATCP［オプション］［ファイル名］

-R：BATCPによって生成されたCOMファイルをバッチファイルに戻す．

-P［Password］：内容を読めない形に変更する．

-E：エコーバックの禁止．

-L：拡張コマンドの使用（後述）．

-I：INT 2EHを使用して起動．

-B［filename［.COM］］：BATCPの規約にあうプログラムを取り込む．
（インクルードできるプログラムとして，BU.COMがある．BU.COMは，BATCP同様に清水洋平氏の作成したフリーソフトウェア，BU.COMには，バッチファイルをより便利に利用するためのユーティリティコマンドがいくつも用意されている．詳しくは，BU104.LZHに含まれるBU.DOCおよびBU.HLPを参照のこと．）

-N：［CTRL］＋［C］による中断を無効にする．

バッチファイルのコマンドの拡張

BATCPは，-Lオプションをつけることで，以下の拡張されたコマンドを含むバッチファイルをコンパイルできる．ただし，拡張コマンドはFOR文内では使用できない．

GOSUB ［Label］～RETURN

［Label］で指定するサブルーチンに分岐する．RETURNコマンドにより呼び出した次の行から処理を続ける．GOSUB～RETURNは，入れ子にすることができる．入れ子は最大8まで可能である．

EXIT

バッチの実行を中断する．この命令以降のバッチは実行せずに終了する．なお，-Iを付けてコンパイルしなければデフォルトで使用できる．

ONGOTO ［Label］ <［Label］...>

ERRORLEVELの値によりGOTOを実行する．ERRORLEVELが0ならばいちばん左のラベルにジャンプし，ERRORLEVELが1増えるごとに1ずつ右のラベルにジャンプする（初期値が0で始まる点に注意）．ラベルの数よりもERRORLEVELが大きい場合は何もしない．

ワンラインNEWS
ブラザー工業はマッキントッシュの文字や画像イメージなどを拡大処理し，大型プリンタに出力するドライバ「デザインスクリプト5000」を開発した．

ONGOSUB [Label] 〈[Label]...〉

ONGOTOとほぼ同じ機能だが，行き先でのRETURNコマンドにより，この命令の次の行に戻れる．

バッチファイルは，単に起動時のAUTOEXEC.BATと，あとからデバイスドライバを組み込むために使うくらいという人が多いだろう．しかしMS-DOSが走るマシンなら，すべてバッチプログラミングのための環境が整っているといえる．しかも，スクリプト次第では十分にツールとしての役割を担うこともできる．なお，弊社刊行の「PC-PAGE 11 MS-DOSバッチファイル」にも，高度にバッチファイルを活用したプログラミングが紹介されている．

■BATCP Ver.1.21

名称：BATCP／分類：コンパイラ（?）／登録名：BATCP121.LZH／作者：清水洋平（Paci,NIFTY-Serve:RGD00165/pcs:pcs35438）／動作環境：MS-DOS ver.3.xの動く機種／登録月日：1991年6月5日／復元方法：LHA E BATCP121／入手場所：ASCII-NET PCS(pool msdos #4017)

■BU Ver.1.04

名称：BATUTY／分類：ツール／登録名：BU104.LZH／作者：清水洋平（Paci,NIFTY-Serve:RGD00165/pcs:pcs35438）／動作環境：MS-DOS ver.3.xの動く機種／登録月日：1991年7月7日／復元方法：LHA E BU104／入手場所：ASCII-NET PCS (pool msdos #4118)

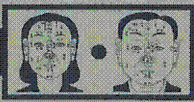
ファイル構成

```
Listing of archive : BATCP121.LZH

  Name          Original    Packed Ratio   Date     Time    Attr Type  CRC
--------------  --------  -------- ------ -------- -------- ---- ----- ----
  BATCP.COM         5457      4116  75.4% 91-06-05 01:21:00 a--w -lh1- F2E2
  BATCP.DOC        11549      5443  47.1% 91-06-05 01:21:00 a--w -lh1- 7853
  BATCP121.DOC      1728       972  56.3% 91-06-05 01:21:00 a--w -lh1- 5C05
--------------  --------  -------- ------ -------- --------
     3 files       18734     10531  56.2% 91-06-14 19:09:58

Listing of archive : BU104.LZH

  Name          Original    Packed Ratio   Date     Time    Attr Type  CRC
--------------  --------  -------- ------ -------- -------- ---- ----- ----
  BU.COM            6198      4823  77.8% 91-07-06 01:04:08 a--w -lh5- 672B
  BU.HLP           13432      5022  37.4% 91-07-06 01:04:08 a--w -lh5- D7F8
  BU.DOC            7290      3631  49.8% 91-07-06 01:04:00 a--w -lh5- 61B2
--------------  --------  -------- ------ -------- --------
     3 files       26920     13476  50.1% 91-07-07 20:48:32
```

HEALTHY?  UNHEALTHY?

⑪ 胃が痛くなる話

精神的葛藤や欲求不満といったストレスは，プログラム作りと切り離せないものだ．趣味で気楽にプログラムを組む人ならいざしらず，締め切りに追われ胃が痛くなる思いで仕事をしているプログラマは，ストレスに取り付かれたような存在である．ストレスはまず胃にくるらしい．「胃が痛くなる思いが」本当になってしまうのだ．

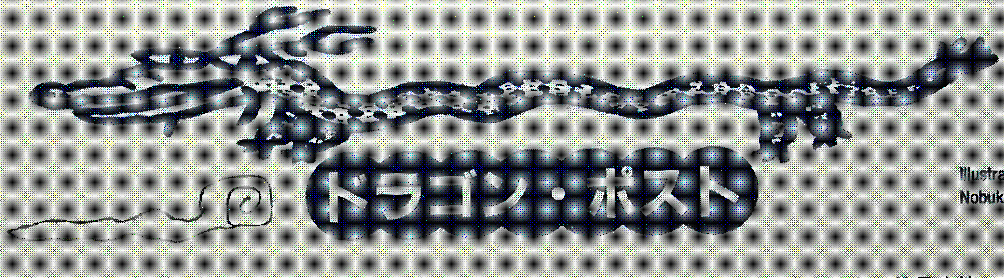
胃が消化器として活動するために重要な胃液は，金属をも溶かしてしまうほどの強烈な酸だ．胃の粘膜は，胃液から自らを守るために，絶えず粘液を出している．ストレスがたまると，胃の粘膜の自己防御活動が弱まり，胃液にやられてしまう．ひどいときには粘膜に穴があいてしまう．これが，胃潰瘍というわけだ．

ストレスの強い仕事をしていると，知らないあいだに潰瘍ができて，知らないあいだに治っていることがある．これが，健康診断のときにひっかかって，再検査になったりする．たいした痛みではないといって，胃のもたれなどを感じても放ってある人は用心したほうがいい．

胃の調子が悪いときに，よく利用される胃散．これには，たいがい炭酸水素ナトリウムが入っている．これは重曹のことで，アルカリによって，胃散を中和する働きがある．中和するときに炭酸ガスが出て，げっぷといっしょに胃の中のガスを出してくれる．それで，胸がスーっとする壮快感があるというわけだ．炭酸マグネシウムや沈降炭酸カルシウムも同じ働きがある．ちなみに，メントールなどはスーっとする効果を出すために配合されることがあるし，苦みや辛みのある薬は食欲を増進させる効果がある．

胃液を中和するからといって，胃散を大量に飲んではいけない．身体が，必要な胃液が足りないと判断して，大量に胃液を分泌させることがあるからだ．

アスキーはソ連のプラウダ紙の中傷記事によって中断していたソ連製ソフトの製品化を再開した．今年12月にもアスキーブランドで売り出す予定．



ドラゴン・ポスト

Illustration
Nobuki Fukuhara

この欄は，読者からのお便りを中心に構成しています．本誌に対するご意見・ご感想などをお待ちしております．鬼より恐い読者からの声ということで，「ドラゴン・ポスト」です．本誌についているアンケートはがきにご意見の欄を設けてありますから，そこに書いてください．ペンネームでも歓迎です．ただし，はがきには本名と住所をしっかり書いてくださいね．掲載誌をお届けします．

★でも，やっぱり386，486マシンがほしい！

286マシンでもある程度は不便を感じませんが，でも遅い．現在のソフトは非常に大きく重くなっているのが現状です．そんなソフトを286マシンで動かすと悲劇です．まったく仕事にならない場合もあります．一太郎・マルチプランV3で，行，列のデリートを行って下さい．終わるまでにお茶できますよ．（植木孝一）

――さすがに8ビットマシンを使っている人はあまりいないようですが，MS-DOSマシンには86，88から486まで，さまざまなCPUが搭載されています．遅いCPUを使っている人は，それなりのソフトを選ばないとつらい世の中になってきました．

★OS/2やWindows3.0のためには……

最近は286どころか，386•486が当然のようになってきましたが，OS/2やWindows3.0のためにはしかたがないのか，考えさせられますが．（藤井孝夫）

――486以上でないとWindows 3.0は使う気にならないと明言している人もいます．がまんできる範囲は人によって違うのでしょうが，Windows 3.0やOS/2はそれなりのCPUやメモリを要求するソフトウェアではあります．

★アー歳はとりたくないなぁ

いまどき！アセンブラしかわからない私は，この世界でも古株になりつつあるんだなぁーと感じます．若干22才！ですでにパソコン歴10年．アー歳はとりたくないなぁ．プログラムのネタにもこまってくるし，キーを打つ手はふるえてくるし，老化現象かな？あーヤダヤダ，だれかこんな私に愛の手を!!（Norida）

――22歳でそんな！わしらはどうすればいいの？でも，子供のころからアセンブラやってるわけだから，「老化現象」はパソコンのタタリだったりして……(^_^;

★ユーザーインターフェースぐらいは早く統一して！

仕事柄，MS-DOS，OS/2，UNIX，大型機用OS，種々雑多なOSとお付合いしていますが，いつも一つの機械からもう一方へと移動するたび，頭を切り換えるのに時間がかかってしまいます．ユーザーインターフェースぐらいは，早く統一してもらいたいものです．（CANCER）

――DOSの世界だけでも，PC-DOS，MS-DOS，DOS/V，98DOS，AX-DOS……などなど，コマンドレベルではともかく，ドライバ類になるとすぐには対応できませんものね．やっかいなのは，それぞれが別のユーザーインターフェースを持っていること．それが統一されるのはいつになることやら……．

★システムがぶっとんだ日

今日は運が悪いというしかない日だった．98NOTEのRAMが午前中に読み取りできなくなり，午後にはハードディスクのシステムがぶっとんだ．ちゃんと電源をつないでおいたのに……なんでこうなるのぉ！バックアップをしておきましょうネ，皆さん！さあ，インストールしなおそうっと．めんどくさいよおー．（角井正彦）

――ハードディスクがとぶ……．読者のお便りでも，身の回りでもよく聞く話です．私たちは，いつまでこういう苦労をしなければならないのでしょうか．ストリーマーやMOが安くなればそれですむ話なのですが．

★日本ではIBM PC関係の情報がすくない

日本では，IBM PC関係の情報が極端にすくない．DOS/Vも出ましたし，またPC-DOSV5.0もDOS/Vに対応しますので，今後は，DOS/VおよびIBM PCについての特集などの記事を載せて下さい．（中村年男）

――DOS/Vの発表以来，IBM PC関係に興味をもたれた方も多いようで，最近こういったご意見が増えています．本誌9月号の西川氏の「486/33MHzマシンが25万円」記事への反響もかなりありました．ちなみに編集部の5人中3人の机にも55/ZなどIBMのマシンが載っています．

★とうとう買ってしまいました

とうとう買ってしまいました数値演算プロセッサを．INTEL387DX-16．速い！ 今までグラフやフラクタルをやっていたとき，画面が完成するまで別のことをやってましたが，これからは画面を眺めていても疲れない．で

も，もっとすごい利用法を知りたい！（山本　光）
——えへへへ．あるんですよね．こういうことって．「すごく速い！気持ちいい！でも，何に使うの？」って．「Windows 3.0を486でぶんまわしている，でも，アプリがない」なんていう人も多いのでは……．

★待っているのは，愛機98と育児です

私がパソコンと付き合いだしたのは今を去るウン年前，当然独身．ところが今は，三児の父．疲れはてて帰宅すると待っているのは愛機98と育児です．パソ通もしたい，プログラムも組みたい．なのに98とむかい会う時間はどんどん短くなっていく……パソコンが育児をする時代はまだ遠いのでしょうか．(ROM)
——お子さんはいくつなのでしょうか？　ディスクドライブに異物を突っ込まれる，キートップがとれてしまう，などといった被害にはまだあっていませんか．

★アセンブラを始めました

最近アセンブラを始めました．難しいといわれていますが，それはいきなりアセンブラの入門書を理解しようとするからです．アセンブラを学ぶ上での第一ステップはCPUの解説書を理解することです．CPUの知識さえあれば，アセンブラほど簡単で親しみやすいプログラム言語はありません．しかも，CPUの知識は忘れようと思っても忘れられないものですし，一石二鳥といえるでしょう．（萩原　聡）
——本誌では，いつも考え，工夫して特集のテーマを選択しているつもりです．でも，一番評判がよかったのは，アセンブラの基礎を取り上げた特集でした．アセンブラの力ってすごいのね．

★OS/2 PMのシステム開発をやっています

今，OS/2 PMのシステム開発をやっています．そして今月の特集でWindowsのプログラミングは取り上げられていたけれどPMはナシでした．すこし残念でした．すでにというか，やっぱりOS/2はもうダメなのですかネ？ちなみにパーソナルOS/2ユーザは少ないとしても，ビジネスユーザは案外いるのではないかと思うし，読者にもその人たちのためのプログラムを作っているプログラマだって結構いると思うのですが……．（佐野浩之）
——WindowsとOS/2が統合されるという時期があったり，別々のコースを進むのだという観測があったり，いずれにしろ，海の向こうのメーカー間の動きが反映しているのですね．

★床屋で，パーマをかけながら？

最近，毎月のように「Programmer's Page」を買うようになってきた．ネットをやっていないので，付録のディスクを目的に購入している．この「Programmer's Page」を読むのは，きまって床屋でである．パーマをかけているのでその間にだいたい読み切ってしまう．もう少し読み物が多いとパーマ屋さんで読むのに楽だが……．（宮川茂俊）
——何と奇特な人でしょう．ありがたいことです．でも，パーマ屋さんから変な目でみられていません？

★プログラミングは男のロマン

貴誌の内容は私のようなビギナーには多少以上に難しいのです．特に最近はアプリケーションが優秀だし，フリーソフトもプロなみなので自分でプログラムを組む必要はないのですが，男のロマンと思い，これからも勉強するつもりですので宜しくお願いします．（寺尾和明）
——「必要はプログラミングの母である」とおっしゃっていた方もいましたが，今使っているアプリケーションに満足していても，敢えてプログラミングに挑戦する，ロマンを追い求める気持ちには，必要かどうかなんて関係ありませんよね．

★三代そろいました！

PC386LSを格安で購入．これで，PC-8801（8ビット），PC286V（16ビット）とあわせて，8，16，32ビットと三代そろいました．しかし…88はもはや昔のゲームができるだけ．そのうち286もそうなるのかしら……．時代の流れはおそろしい．個人で流れにのって行くには¥¥¥¥………．（武田徹也）
——NEWマシンとはうらやましい．でも約10年の間に3台目（3代目）とは，ホント，時代の流れはおそろしいですね．やっぱり，四代目は「486マシン」（32ビットだけど……）でしょうか．

★クイズ・ランダムスクリーンの正解に枠がないのは？

相変わらず，記事の内容はわからないのですが，はじから読んでいます．ところでクイズ・ランダムスクリーンの絵は，完成すると分解した画像の枠は消えるのでしょうか？　いつも，完成したつもりでいても枠はあるのに，次号の解答には枠がありません……．（K）
——解答の写真に枠がないのは，分解する前のものを撮影しているからなのです．完成したら「Congratulations!」と表示してほしいなどと，担当者にお願いしているのですがなかなか手がまわらなくて……．

（敬称略）

付録ディスクのご案内

このフロッピーディスクに含まれる実行ファイル，ソースリストおよびテキストファイルは「月刊Programmer's Page」1991年11月号に掲載されたものです．

付録のフロッピーディスクは，5"2HD（8セクタ／トラックMS-DOS）でフォーマットされています．3.5"や2DDなどへのコンバートサービスは行っておりません．なお，オンラインサービスSE-NET（03-5276-1297）でも，同じ内容のファイルを用意しています．

フロッピーディスクの内容は次の通りです．本誌およびディスクに含まれるドキュメントファイルをよく読んでからご利用ください．

■ディスクのおもな内容

●付録ディスクについての説明

ディレクトリ名：ルート

収録ファイル名：

README.BAT（ディスク内容の説明を読むためのバッチファイル，PC-9801専用）

README.DOC（ディスク内容の説明，MS-DOSテキストファイル）

ONLINE.BAT（オンラインソフトの説明を読むためのバッチファイル，PC-9801専用）

ONLINE.DOC（収録されているオンラインソフトの簡単な説明）

V.EXE（ミニページャー，PC-9801専用）

BOOK.TXT（翔泳社書籍目録）

●特集「Windowsプログラムの作り方」

①「Windows APIはこうして使う」（p.2）

ディレクトリ名：TOKU

収録ファイル名：

TOKU1.LZH

PPAGE.LZH（本誌1991年2月号の「特集・Windowsの時代である！」に掲載された「Windows基本プログラム大解剖・永井知己」を再編集したもの）

WINAPI.TXT（本誌1991年2月号の綴じ込み付録「MS-Windows 3.0関数一覧」を再編集したもの）

②「Borland C++」（p.21）

ディレクトリ名：TOKU

収録ファイル名：TOKU2.LZH

③「Turbo Pascal for Windows」（p.41）

ディレクトリ名：TOKU

収録ファイル名：TOKU3.LZH

④「VisualBASIC」（p.51）

ディレクトリ名：TOKU

収録ファイル名：TOKU4.LZH

●プログラム『遊』工房（p.68）

「SNAKE」ゲーム

ディレクトリ名：PLAY

収録ファイル名：PLAY.LZH

●MS－DOSツール・ラボ（p.82）

起動記憶プログラム「TIMERING.EXE」

ディレクトリ名：TOOL

収録ファイル名：TOOL.LZH

●実践C塾（p.92）

ディレクトリ名：JUKU

収録ファイル名：JUKU.LZH

●98活用アセンブラ技法（p.102）

ディレクトリ名：98ASM

収録ファイル名：98ASM.LZH

●綴じ込み付録

ディレクトリ名：DATA

収録ファイル名：WININI.TXT（MS-DOSテキストファイル）

●Programmer's Program（p.138）

ディレクトリ名：PPROGRAM

収録ファイル名：

GCLR.LZH（サンプルとして使用したグラフィック画面をクリアするプログラム・ソース／実行ファイル）

TEST.LZH（サンプルの実行ファイル（GOFF.COM）を，ZZ386で逆アセンブルした結果・ソースファイル）

●オンラインソフト採集帳（p.140）

本誌今月号のオンラインソフト採集帳で紹介したバッチファイルをCOMファイルに変換するコンパイラ（？）「BATCP.COM」と，バッチファイルのコマンドを拡張するユーティリティ「BU.COM」が収録されています．また「.LZH」という拡張子をもつ圧縮ファイルを復元するのに必要な，高圧縮復元書庫管理ソフト「LHA」も収録されています．

ディレクトリ名：ONLINE

収録ファイル名：

BU104.LZH（BU.COM，MS-DOS汎用）

BATCP121.LZH（BATCP.COM，MS-DOS汎用）

LHA213.EXE（LHA Ver.2.13，MS-DOS汎用）

●クイズ・ランダムスクリーン（p.146）

クイズ解読ツール「PUZLE」

本誌掲載のクイズ・ランダムスクリーンの画像をディスプレイ上に再現し，マウスを使って並べ替えることにより，解答を得ることができるユーティリティです．PC-9801シリーズ専用です．詳しい使い方はPUZLE.DOCを参照してください．

ディレクトリ名：OMAKE

収録ファイル名：

PUZLE.EXE（解読ツール，PC-9801専用）

PUZLE.DOC（PUZLEの説明）

CONFIG.SYS（PUZLE用CONFIG.SYS）

AUTOEXEC.BAT（PUZLE用AUTOEXEC.BAT）

RNDSCREN.LZH（画像データ）

TRAIN.LZH（10月号の解答データ）

TRAIN.BMP（Windows 3.0の壁紙）

※使用上の注意

本誌に掲載されたプログラムおよびデータを利用するには，MS-DOSの基礎知識が必要です．このディスクには，MS-DOSのシステムは入っていません．別途，MS-DOSが必要となります．また，ソースファイルのコンパイルなど，別途コンパイラやユーティリティが必要な場合があります．

■圧縮ファイルについて

ディスクスペースを有効に使うために，多くのファイルは圧縮した上で収録しています．「.LZH」という拡張子のついたファイルは，高圧縮書庫管理ユーティリティ「LHA」によって圧縮されたファイルです．これらのファイルを利用するには，まずLHAで復元する必要があります．LHAの利用方法については，ONLINE.DOCをご覧ください．

バーチャルスペースガイド

P子　某ソフトハウスで企画営業を担当している。26歳。

C平　某メーカーのシステムソフトウェア開発部門に所属。25歳。

P子　「ターミネーター2」おもしろかった。前作とは違うアイデアがいっぱいあって，やっぱり，これでもか，これでもかっていう感じで，すごいの。でも，混んでてね，平日でも指定席までいっぱいだったもの。

C平　ほう，このあいだ会社に電話したら休みだっていってたけど，そういうことをしてたわけ。

P子　なによ，その顔は。私なんか，有休が山ほどたまってるんだから。

C平　たまった有休も消化できない僕は，「前作」をまた借りてきて見たりしているのでした。でも，ひさしぶりに見るとやっぱりすごいね。もう終わりだろう思うときが何度かあるけど。

P子　そのたびに，ターミネーターが動き出すというわけね。「前作」では，やっぱりトラックが燃えて，やっとやっつけたという感じで2人が駆け寄るシーンよね。そしたら，炎の中からアレが立ち上がるの。私なんかもう，ギャー！ていっちゃたもの。

C平　聞いた話では，「2」では形状記憶合金のロボットなんかが出てくるんだって？

P子　ほほっほ。あれも新しいアイデアよね。教えてあげようか？どうせ，レンタルビデオに入るまで見れないんでしょう。

C平　教えて欲しいけど，やめとこ。あの映画は驚きが減ったら，その分損するみたいなものだから。僕だって，映画館でみたいという気持ちはあるわけよ。でも，平日の夜遅くはやってないし，休日は混んだ映画館に行く気力がないし……。

■ターミネーター (1984・米)

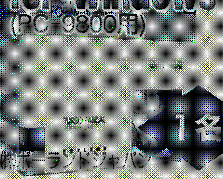
監督：ジェイムズ・キャメロン
出演：アーノルド・シュワルツェネッガー

機械が支配する未来からやってきた殺人マシンが，未来の世界で人類を指導する人物の母となる女性を狙う。それを阻止しようとする未来の戦士。物語は単純だ。SFとしてのアイデアも新しくない。しかし，並外れたバイオレンスと，巧みな展開で見るものをぐいぐい引っ張っている力はすごい。まさに映画的な面白さがある。

アスキー／ポニーキャニオン　107分／3,800円

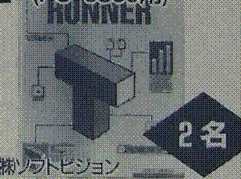
読者プレゼント

1　Turbo Pascal for Windows (PC-9800用)



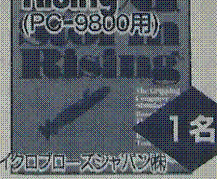
1名　提供：㈱ボーランドジャパン

2　TYPE RUNNER (PC-9800用)



2名　提供：㈱ソフトビジョン

3　Red Storm Rising (PC-9800用)



1名　提供：マイクロプローズジャパン㈱

4　Windows 3.0 特別集中講義

3名　翔泳社・PC-PAGEシリーズ新刊

応募は本誌に添付のアンケートはがきをご利用ください。(1991年11月18日消印有効)。なお，応募者多数の場合は，抽選になります。当選者発表は発送をもって代えさせていただきます。

クイズ・ランダムスクリーン

ある1枚の画像を64分割し，ランダムに並べ替えてあります．
さて，もとの画像は何を写したものでしょうか．

ヒント：パリにある．○○○○○○寺院

正解者の中から30名様に本誌表紙イラスト（日暮修一画）入り月刊プログラマーズ・ページ特製オリジナルマグカップをさしあげます．

応募方法：本誌に添付のアンケートはがきをご利用ください．

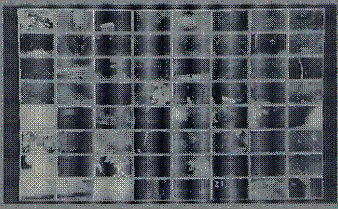
締め切り：1991年11月18日（消印有効）

なお，当選者の発表は発送をもって代えさせていただきます．

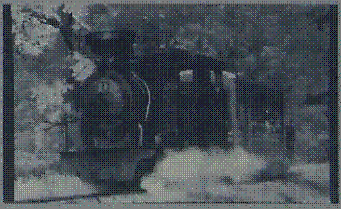
＊画像データは，付録ディスクやBBSでもサービスしています．

10月号の問題と正解

【問題】



【正解：機関車】



月刊プログラマーズ・ページ オンラインサービス（SE-NET）のお知らせ

SE-NETでは，創刊号からの掲載プログラムとオンラインソフトのサービスを行っています．また，「ドラゴン・ポスト」や「編集部への質問」，「プログラマ内輪話」などのコーナーも設けております．読者のみなさまと編集部とのコミュニケーションの場としてもご利用ください．

●BBSへの接続

SE-NET　03-5276-1297

①接続するとIDの入力を求めてきますので，GUESTと入力しリターンキーを押してください．
②お知らせを表示後，メインメニューのコマンドプロンプトが表示されます．
③月刊プログラマーズ・ページへのご意見などは「ドラゴンポスト」ボードに書き込んでください．
④ファイルの転送が終了したら，0と入力して回線を切ります．

Programmer's Page MONTHLY MAGAZINE FOR PROGRAMMERS

月刊プログラマーズ・ページ バックナンバーのご案内

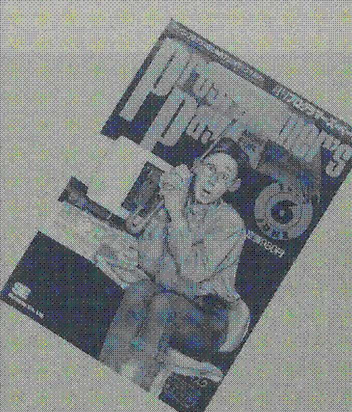
■定価・各980円

（1991年1月号「特集・プログラマの仕事場」・2月号「特集・Windowsの時代である！」・3月号「特集・この言語を使いなさい!!」・4月号「特集・C読本」は品切れです）

1991年5月号 5インチFD付き

特集・基礎力としてのアセンブラ

プリミティブなプログラミング言語でありながら，それ故に無視することができないアセンブラ，コンピュータを理解し，その能力をひきだすためにもう一度アセンブラをみなおす．



1991年6月号 5インチFD付き

特集・研究「プログラミング統合環境」

今流行のエディタ，コンパイラ，リンカ，デバッガを統合した開発環境について，その歴史的系譜から最新のシステムまで徹底的に検証する．

1991年7月号 5インチFD付き

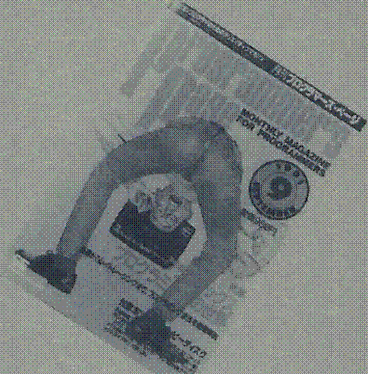
特集・最新「デバイスドライバ」研究

パソコンの拡張性を支えるソフトウェアインターフェースであるデバイスドライバ作成の技法と特徴をMS-DOS，Windows，OS/2の各環境ごとに解説する．

1991年8月号 5インチFD付き

特集・はじめてのオブジェクト指向プログラミング

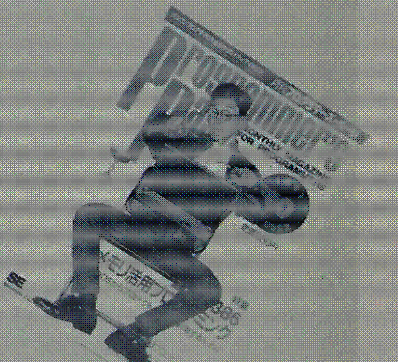
90年代のプログラミングパラダイムとして注目される"オブジェクト指向"の歴史的な発展の過程から基礎概念，分析・設計の方法までを網羅的に解説する．



1991年9月号 5インチFD付き

特集・グラフィックス プログラミングの世界

グラフィックスプログラミングの基礎からフレームバッファの活用，レイトレーシング，Windows，IBM PCのグラフィックスなど，その技法を徹底的に解説する．



1991年10月号 5インチFD付き

特集・286/386 メモリ活用プログラミング

バンク式メモリ，EMS，XMSからプロテクトモードの活用まで，MS-DOSがもつ640KBの壁を破るための数多くのプログラミングテクニックを解説する．

「月刊プログラマーズ・ページ」のバックナンバーは，ご注文いただければ，全国どこの書店でもお求めになれます(書店に届くまで1か月ほどかかります)．小社に直接ご注文なさるときは，何月号かを明記の上，現金書留にて雑誌価格＋送料を下記までお送りください．送料は1部81円です(ただし7月号は1部86円)．2冊以上の場合は，号数・冊数により送料が異なりますので，小社までお問い合わせください．

〒102 東京都千代田区平河町
2-4-14 平河町KSビル4階
(株) 翔泳社
マーケティング部PP係宛
TEL 03-3263-0447

Programmer's Page

MONTHLY MAGAZINE FOR PROGRAMMERS

月刊**プログラマーズ・ページ12月号**

(11月18日発売)

付録：5″ 2HDフロッピーディスク付き!!

特集

やっぱり僕らはBASIC!!

ひと昔前のROM-BASICとは異なり，現在のすすんだBASICは，構造化プログラミング，マルチランゲージに対応した，強力なプログラミング言語に生まれ変わっている．MS-BASIC 7.1，TrueBASIC，QuickBASIC，BASIC/98，さらにVisualBASICまで，プログラミング言語BASICの再認識を図る．

月刊プログラマーズ・ページ　1991年11月号
1991年11月1日発行

発行所　株式会社翔泳社
東京都千代田区平河町2-4-14平河町KSビル4F・5F
TEL 03-3263-0447　〒102
発行人　長廻健太郎
編集長　清水　隆
副編集長　鈴木嘉平
編集スタッフ　杉野貴美子＋上田純美礼＋芳山充弘
デザイン　SE（翔泳社デザイン研究所）
広告スタッフ　相澤正夫＋澤本良知＋種村嘉彦
協力　根本裕三郎＋(株)ドモン・マインズ＋佐藤裕美＋小林弘人（図版デザイン）
写植　株式会社東巧社＋株式会社トーヨー・プリント社
印刷　凸版印刷株式会社
定価　980円（本体951円，税29円）

© 1991 SHOEISHA
本書の一部または全部を著作権法の定める範囲をこえ，無断で複写，転載，テープ化，データファイル化することを禁じます．
＊本書の内容に関するご質問は，ご面倒でも必ず文書にて編集部までお問い合わせください．

EDITOR'S PAGE

●5年以上も使っていたビデオデッキが，豚児に壊されてしまった．仕方がないので，SVHS＋BSというデッキを買ってしまった．うちのアンテナ端子には，BS信号も入っているらしい，どうしたら映るのかと調べてみると，分波器が必要だという．なんだ，家電もけっこう面倒になっているじゃないか．こういう面倒な機械を使っている人が，パソコンとなると，フロッピーディスクのコピーさえままならなかったりする．前から思っているのだが，教習所に通わなければならないほど面倒な，車の運転をできる人は日本の人口の半分もいるのに，カメラの操作がわからなかったり，パソコンがまるっきりダメだという人が多いのはなぜだろう？（TA）

●最近，SuperSessionというsharewareの通信ソフトを手に入れた．韓国製のソフトなのだが，その洗練されたデザインと使い勝手のよさは目を疑うほどだ．韓国製であるからハングルに対応しているのは当然として，なんとDOS/Vにも対応している！　本家であるはずの日本では，まだろくにソフトが出ていないというのに……．なんだか情けなくなってしまうなぁ．（嘉平）

●先日，健康診断がありましたが，事前に記入する問診票に，「テレビを見てもおもしろくない」「人生がつまらなく感じる」「将来が不安」などのチェック項目が，たぶん潜在的なストレスのチェックなのでしょうが，改めて考えると，何となく当てはまるような，そうでもないような．「ストレスを感じない＝無神経」と思われそうですが，私の場合，意識せずに突っ走るのがストレスを貯めないコツかもしれません．（ま）

●毎日まいにち雨の日ばかりだというのに，傘をなくしてしまいました．持っていると雨に降られない不思議な傘だったのに．何がたいせつなのかはなかなかわからないのに，なくす時はあっけないほど簡単なものだというのは，何事にも言えることなんでしょうね．（kos）

●ハワイで優雅な(？)休みを過ごしにでかける．鞄の中には，短パンとTシャツ，そして98note．ハワイにまでパソコンをもっていくなんて，僕ってほんとに(を)．（Yoshi）

●お知らせ

9月10日ころから1週間ほど，SE-NETへのアクセスができなくなっていました．原因は，電話回線の混線で，コンソールからはログインできるのに，電話回線からは接続できませんでした．回線は修復いたしましたので，現在は通常通り動作しております．ご迷惑をおかけした方々にお詫び申しあげます．

愛読者カードご記入のお願い

本誌をお買い上げいただき、誠にありがとうございます。
お手数ですが、今後の誌面づくりの参考にさせていただきたいと存じますので、
各欄にご記入の上、ご返送くださるようお願い申しあげます。

月刊プログラマーズ・ページ READER'S SURVERY CARD 愛読者カード

本誌の号数　　年　　月号

本誌を知った理由
□1.書店店頭　□2.他社出版物(　　　)
□3.小社出版物　□4. DM　□5.口コミ

パソコンの所有形態(機種名)
□1.個人所有　(　　　)
□2.職場にある(　　　)
□3.持っていない

本誌を購入した書店　　市(区)　　書店

よく使用するソフトウェア

よく使用するOS

今月号でおもしろかった記事(いくつでも)

今後取り上げてもらいたい特集のテーマ

貴方が選ぶベスト３

パソコン雑誌	①	②	③
ソフトウェア	①	②	③
パソコン本体	①	②	③

本誌についてのご感想、お気づきの点、その他どのようなことでも結構ですので、ご意見をお聞かせください(「ドラゴン・ポスト＝読者の声」欄に掲載させていただく場合もあります。ペンネームをご希望の方はあわせてご記入ください)。

月刊プログラマーズ・ページ 広告資料請求カード

8	会社(学校)名		9	所属(学部)		10	従業員数	

※下記の該当する番号をご記入ください。

11	業種		12	職種		13	パソコン利用状況	

業種　①製造業　②金融・保険・不動産業　③商社・卸・小売業
④建設業　⑤通信・運輸　⑥システム・ソフト
⑦その他サービス業　⑧公務員　⑨その他

職種　①コンピュータ(ハード)技術者　②コンピュータ(ソフト)技術者　③エレクトロニクス技術者
④プログラマ　⑤その他の技術者　⑥専門職
⑦事務・営業職　⑧経営管理職　⑨商工サービス
⑩教職　⑪学生　⑫その他

パソコン利用状況　①会社で利用　②個人で利用　③会社でも個人でも利用　④利用していない

●希望される広告資料　　年　　月号

広告資料請求No. / 希望項目												
A 購入希望	A	A	A	A	A	A	A	A	A	A	A	A
B 説明希望	B	B	B	B	B	B	B	B	B	B	B	B
C 資料希望	C	C	C	C	C	C	C	C	C	C	C	C

定期購読のご案内

綴じ込みの振替用紙を使って、本誌の定期購読をお申し込みになれます。
1年間12冊=11,760円(本体11,417円/税343円)
郵送料サービス

郵便はがき

料金受取人払
麹町局承認
3908
差出有効期間
平成4年11月
30日まで
切手不要

102-00

東京都千代田区平河町2-4-14
平河町KSビル
株式会社 翔泳社
マーケティング部 行

月刊プログラマーズ・ページ
READER'S SURVEY CARD　愛読者カード

氏名	性別 □1男 □2女	生年月日 19　年　月　日
住所　〒		電話 (　　) −

職業			ユーザー区分
□1ソフト技術者	□2ハード技術者	□3技術系会社員	□1プロ
□4事務系会社員	□5営業系会社員	□6教師	□2マニア
□7一般公務員	□8経営管理職	□9理系学生	□3一般ユーザー
□10文系学生	□11その他(　　)		

プレゼント記入欄	クイズ・ランダムスクリーン回答欄
希望商品番号	答

郵便はがき

料金受取人払
麹町局承認
3910
差出有効期間
平成4年11月
30日まで
切手不要

102-00

東京都千代田区平河町2-4-14
平河町KSビル
株式会社 翔泳社
月刊プログラマーズ・ページ
広告資料請求係行

＊当社で記入

1	SE	2	受付No.	

月刊プログラマーズ・ページ広告資料請求カード

3	会員番号	

＊すでに会員番号をお持ちの方はご記入ください。

4	フリガナ 氏名	姓　　名	5	生年月日	19　年　月　日

6	自宅住所	〒　電話 (　　)
7	会社住所	〒　電話 (　　)

資料送付希望先の番号に○印をつけてください。

払込票

通常払込料金加入者負担

口座番号 東京 4-154493

加入者名 株式会社 翔泳社

金額 11760

払込人住所氏名 ※

備考

受付局日附印

切り取らないで郵便局にお出しください。

記載事項を訂正した場合は、その箇所に訂正印を押してください。

払込通知票

通常払込料金加入者負担

口座番号 東京 4-154493

金額 11760

加入者名 株式会社 翔泳社

料金 払込み 特殊 円

備考

受付局日附印

払込人住所氏名 ※（郵便番号　　　）

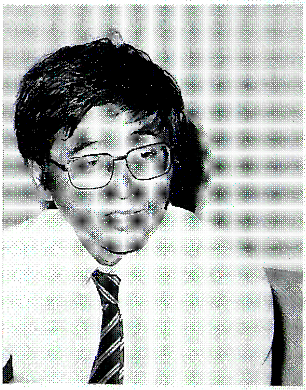
TEL（　　　　　）

この払込通知票は、機械で使用しますので、下部の欄を汚さないよう特に御注意ください。また、本票を折り曲げたりしないでください。（郵政省）

各票の※印欄は、払込人において記載してください。

ム株式会社

プのシステムインテグレータとし
ア企業である。その守備範囲は広
ージソフトの分野までをカバーし
事が主な業務のため、一般には
ログラマの世界では、フルスクリ
f」を開発した会社、と言ったほ
。
部次長池田実氏、基本ソフト開
務部採用担当主査友田昭博氏を
や人材採用についてお話をうかが



部次長　池田実氏

本版スペースシャトル＝HOPE計画に参加し、熱制御系の設計解析をモデルを担当している。これは機内の温度を一定に保つ冷却装置の設計解析を行うものだ。

エネルギー分野では、電力設備を広域ネットワークで結び、電力の効率的な供給を実現する、AI応用の電力制御システムの開発などが代表例といえる。

また、ファジィ理論に基づく高度のAI的シミュレーションを駆使した「バスダイヤ自動編成システム」ほか、交通・航空管制システムにも実績がある。

入れはじめている。

パッケージソフトにも意欲的

従業員数約530名、年間売上高約115億円と、大手ソフトウェア企業のひとつに数えられ、業務的にも大規模なシステム開発を中心としている同社が、このところパッケージソフトへの取り組みを積極化させている意味は何だろうか。開発

田氏はこう語る。

「パソコンがこれだけ社会に浸透し、会社でも家庭でも使われる時代です。その中で、当社もソフトウェア企業として、独自のパッケージソフトを世に問うことは大切なことだと考えています。

それはまた、社内的に見えても、プログラマー人ひとりにとって、いい刺激になっていると思いますよ。　プログラマなら誰だって、会社や家庭で多くの人たちに使ってもらえるソフトを作ってみたいじゃないですか。

また、パッケージソフトを通じて、広く当社を知っていただける、というメリットも

定期購読のご案内

- 定期購読は、1年間12冊と限定させていただきます。
- 購読料のお支払は、この振替用紙をご利用ください。銀行振込、現金書留によるご送金は、ご遠慮ください。
- 振替用紙に、ご住所、お名前をご記入のうえ、郵便局の貯金窓口にてお支払ください。
- 小社より領収書の発行はいたしませんので、郵便局にて「郵便振替払込金受領証」をお受取ください。
- お申し込みいただいた方には、弊社より直接発送いたしますので、到着が発売日より遅れることがございますが、ご了承ください。

この欄は、加入者あての通信にお使いください。

月刊プログラマーズ・ページ　定期購読直接申し込み書

Programmer's Page MONTHLY MAGAZINE FOR PROGRAMMERS

年　　月号より

年間購読料 11,760 円(税・送料込み)

■ご希望の購読開始月号をご記入ください。

■送付先（住所・氏名は正確にご記入ください）

住所	□□□-□□	フリガナ	
		氏名	
		TEL	

この払込通知票は、機械で使用しますので、下部の欄を汚さないよう特に御注意ください。また、本票を折り曲げたりしないでください。　（郵政省）

郵便はがき

102-00

東京都千代田区平河町2-4-14
平河町KSビル

株式会社 翔泳社
月刊プログラマーズ・ページ
広告資料請求係行

2	受付No.	

マーズ・ページ広告資料請求カード

＊すでに会員番号をお持ちの方はご記入ください。

名	5	生年月日	19　年　月　日

綴じ込みの
1

□7 一般事務員　□8経営管理職　□9理系学生　□0 一般ユーザー
□10文系学生　□11その他(　　　　)

プレゼント記入欄	クイズ・ランダムスクリーン回答欄
希望商品番号	答

6	自宅住所	〒　　電話　(　　)	資料送付希望先の番号に○印をつけてください。
7	会社住所	〒　　電話　(　　)	

Programmer's Page

広告企画

■

ちょっと気になるこの会社

日本システム株式会社

日本システムは、東芝グループのシステムインテグレータとして躍進する、大手のソフトウェア企業である。その守備範囲は広く、スペースシャトルからパッケージソフトの分野までをカバーしている。システム開発関係の仕事が主な業務のため、一般にはあまり知名度は高くないが、プログラマの世界では、フルスクリーンエディター「日本語Brief」を開発した会社、と言ったほうが分かりが早いかもしれない。

東京・新宿区の本社に、開発部次長池田実氏、基本ソフト開発第2部主査緒方正典氏、総務部採用担当主査友田昭博氏を訪ね、同社のソフトウェア開発や人材採用についてお話をうかがった。

大規模プロジェクトを柱に

東芝系列のソフトウェア企業ということもあって、日本システムが手がけるソフト開発には大きなプロジェクトが多い。

開発の柱として、「ネットワーク」「エネルギー」「先端技術（ファジィ・AI・宇宙）」の3大テーマが掲げられているのを見ても、同社の意欲と自信のほどがうかがえる。

大規模プロジェクトの例としては、まず宇宙関連がある。スペースシャトルや人工衛星などの解析・シミュレーション、さらにNASDA（宇宙開発事業団）の日本版スペースシャトル＝HOPE計画に参加し、熱制御系の設計解析をモデルを担当している。これは機内の温度を一定に保つ冷却装置の設計解析を行うものだ。

エネルギー分野では、電力設備を広域ネットワークで結び、電力の効率的な供給を実現する、AI応用の電力制御システムの開発などが代表例といえる。

また、ファジィ理論に基づく高度のAI的シミュレーションを駆使した「バスダイヤ自動編成システム」ほか、交通・航空管制システムにも実績がある。

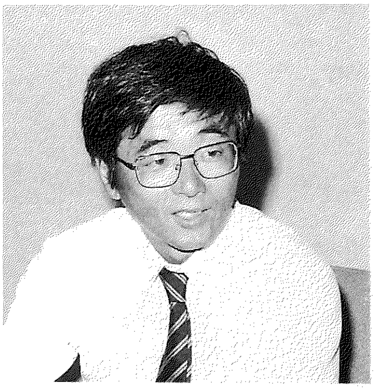
その他、気象通信システム、商社ネットワーク、新聞ネットワークなどのネットワークシステムや銀行向け外国為替システムの開発など、同社のプロジェクトには、きわめて社会性の高いシステムの開発が多いのが特徴だ。

同社では、これらの受託システム開発が全体の90％を占めるという。

その一方で注目しておきたいのが、パッケージソフトの開発・製品化への展開だ。プログラマには定評のあるプログラミング用エディター「日本語Brief」、社内で開発から手がけた建築CAD「Bi-CAD」、そして速読法を身につけるためのトレーニングソフト「ダイナブック速読」など、個性と実力を備えたソフトづくりをめざし、パッケージソフトの分野にも力を入れはじめている。

パッケージソフトにも意欲的

従業員数約530名、年間売上高約115億円と、大手ソフトウェア企業のひとつに数えられ、業務的にも大規模なシステム開発を中心としている同社が、このところパッケージソフトへの取り組みを積極化させている意味は何だろうか。開発部の池田氏はこう語る。



開発部次長 池田実氏

「パソコンがこれだけ社会に浸透し、会社でも家庭でも使われる時代です。その中で、当社もソフトウェア企業として、独自のパッケージソフトを世に問うことは大切なことだと考えています。

それはまた、社内的に見えても、プログラマ一人ひとりにとって、いい刺激になっていると思いますよ。 プログラマなら誰だって、会社や家庭で多くの人たちに使ってもらえるソフトを作ってみたいじゃないですか。

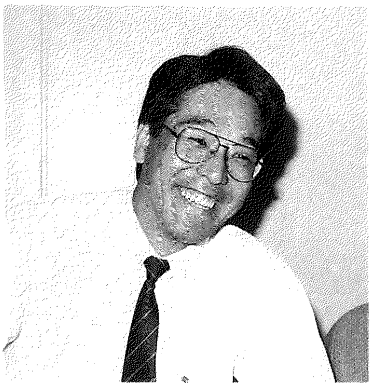
また、パッケージソフトを通じて、広く当社を知っていただける、というメリットも

ありますね。」

社内ではまだまだ小さな部門ではあるが、パッケージソフトの開発には夢がある。エディター、建築用CAD 、トレーニングソフトの次には、どんなソフトが登場してくるのだろうか。開発に携わる緒方氏は「やっぱり、売れるソフトを作りたいな」と笑いながら、こんなことを語ってくた。

「内容的にしっかりしたものを作っていきたいという気持ちはありますね。会社も開発に関して自由にやらせてくれますし、アイデア倒れに終わるようなものではなく、長くユーザーに愛されていくような製品を開発していくことが大切だと思っています。」

新人研修の内容に自信あり



基本ソフト開発第2部主査
緒方正典氏

さて、採用について見てみよう。同社の採用実績は、昨年度66名、今年度72名。中途採用者は少なく、大学・専門学校からの新卒者がほとんどを占めているという。したがって、新入社員が重要になってくるわけだが、新卒者を一人前に育てることについては、同社はかなり自信を持っているようだ。採用担当の友田氏に聞いてみた。

「新入社員教育として6ヵ月間のの研修を行っています。6ヵ月の研修期間というのは、最近は多くの会社で実施しているので珍しくはありませんが、その内容ということになると、会社によってマチマチだと思います。会社の中できちんと教育を行っていくというのは、実際にはなかなか難しいことなんですね。つい"習うより慣れろ"で仕事の処理が優先してしまい、教育的な観点が抜け落ちてしまう傾向にあります。

そこで当社では、新人社員教育を行うために先輩社員による教育実行委員会をつくり、専任の教育担当を置いて、計画・達成・確認に当たっています。

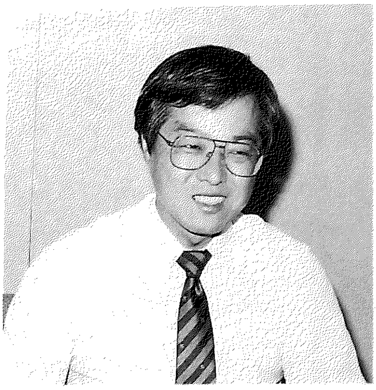
新卒者の30%はコンピュータの素人ですが、大丈夫ですよ。半年の研修を終える頃には、すっかり会社に溶け込んで仕事をしています。」

というわけで、新人社員教育や、それ以後の能力開発については、全社的な体制が敷かれている。さすがに体力のある会社は違うな、という印象だ。

プログラマーの情熱と誇り

ソフトウェア業界では、いま、プロジェクト管理などを任せられるマネジメント能力を備えた人が求められている、といわれる。では、同社の求人像はどうだろうか。

「即戦力としてそういう人がいれば、当社でもほしいところですが、なかなかそういう人がいないのが現実。やはり、育てていくことを考えるのが第一じゃないでしょうか。ですから、技術がどうこうというより、人間として一緒に楽しく仕事のできる人なら、歓迎です」と友田氏。



総務部採用担当主査
友田昭博氏

また池田氏は「私はプログラマという技術職は創造的だし、奥の深い仕事だと思います。アメリカではお歳を召されていてもプログラマとして現役で活躍されている方がおられますよね。日本でも、プログロマという仕事に情熱と誇りを持ち続けていける環境というのが必要だし、若い人たちには、そういうプログラマ像をめざしてほしいと思うんですよ」と語る。そして、緒方氏も「パソコン大好きという人、自分もプログラムを作ってみたいという人、そんな人にきてほしい」と語るのである。

こうして話はいつしか"プログラマ談義"に……。みなさん、プログラマという仕事に情熱と誇りを持っていらっしゃる。この仕事が大好きなのである。こちらも初心に帰って、"コンピュータとは、人間の夢を実現するための道具なのだ"ということを改めて確認させられた次第。率直で、肩ひじ張らないよう話ぶりが、そのまま同社の雰囲気を語っているように感じられた。

高水準の技術開発にはパーフェクト精神が根幹です。

ワンランク上の提案型ソフトウェアハウス＝エルテックスは、ユーザーや上司の求めている以上の開発を追求していく、我がままプロ集団の組織。常に高レベルの開発を行っていくためには、たえず外部の開発にアンテナを張りめぐらしておかなくてはなりません。その一環として、毎年数名、ラスベガスのコムデックスショー等、1週間程度の視察旅行に参加します。パーフェクトなものを求め続けていく技術者たちに、私たちは、たえず評価を与えていきます。

技術者随時募集

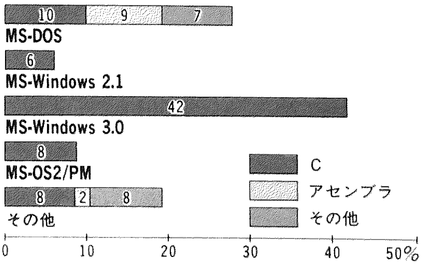
会社概要　ELTEX

設　立	昭和60年12月14日
資本金	1,000万円
事業内容	パソコンを中心としたシステムの提案・開発・販売　①CD-ROM・LD　など新しいメディアに対応したシステムの開発 ②ネットワークの端末システムの開発 ③特定業種向けCADの開発・販売 ④基本ソフトの開発
特　徴	大手クライアント向けのシステム開発が多く、ネットワーク関連システムやマルチメディアなど幅広く対応している。また、ＣＡＤを中心とした自社プロダクツにも積極的で、その比率も増加している。基本ソフトの分野では、ハードウェア専門のベンチャー企業とジョイントし、ＯＥＭ製品のＢＩＯＳ・ユーティリティなどを企画から手掛けている。
取引先	三菱電機、三菱自動車、日本航空　他
関連会社	株式会社ソーテック、学校法人福井情報ビジネス総合学院

募集要項　ELTEX

職　種	①システムエンジニア　②プログラマー
資　格	専門卒以上　20～36歳の方　※Ｃ言語・アセンブラ言語経験者大歓迎　※未経験者でも好奇心・向学心の旺盛な方は歓迎します。
給　与	17万円（専門卒初任給）～46万円（36歳課長、諸手当含む） 24歳例：23万5000円（情報処理１種手当・住宅手当含む） 30歳例：31万6000円（中級SE、情報処理１種・住宅手当含む）
手　当	時間外手当、役職手当（5000円～６万円）、住宅手当（１万5000円）、資格手当（２種：5000～特種：２万円）、通勤手当
賞　与	賞与年２回（平成１年度実績年間５ヵ月）、決算賞与年１回（平成１年度実績１ヵ月）
勤務地	本社（横浜駅近く） ◇派遣・出向はありません。
勤務時間	フレックスタイム制（コアタイム10:00～16:00）試用期間中は標準時間9:00～17:45
休日休暇	完全週休２日制（土・日）、祝日、年３回連休（GW・夏期・年末年始）、有給・特別・休暇
福利厚生	各種社会保険完備、退職金制度、財形貯蓄、社員旅行、海外研修、定期健康診断、社員寮、スポーツクラブ法人会員
応募方法	電話連絡の上、履歴書(写真貼付)、職務経歴書を持参し来社ください。郵送も受け付けます。
連絡先	〒220 横浜市西区南幸二丁目20番５号　東伸24ビル５階　℡(045)322-9667　総務課

最近のOS・言語別開発動向



採用HOT・LINE
TEL.(045)322-9667

株式会社エルテックス

米国「The C Users Journal」「TECH Specialist」提携誌

月刊 Cジャーナル

C言語を主体とするコンピュータプログラマのための月刊情報誌

11
DEC.1991

C JOURNAL

定価980円(本体951円)

11月号 10月8日発売

●連載●

★STANDARD C
★Dr.C's Pointers
★Question & Answers
★Programming Style 第4回
★MINIX通信 第10回
★たのむぜ／DOS/V／ その4
★グラフィックユーザインタフェイス環境 第4回
★MacのクラスライブラリとGUI 4
★Pascal言語系の制作 第2回
★特別寄稿 D.Thielenの"No-Bugs"

第一特集
Windowsプログラミング

第二特集 **CD-ROM**

付録DISK(5"2HD)
●Pascal-B
コンパイラ及び全ソースコード
●掲載記事の全コード収録

12月号
11月8日発売

特集 アセンブラ &
ディスクアセンブラ

株式会社ビレッジセンター 出版局
〒101 東京都千代田区三崎町3—1—16 神田アメレックスビル2F
●営業部:03-3221-1825●編集部:03-3239-1950

C JOURNALは毎月8日全国の書店でお求めになれます。

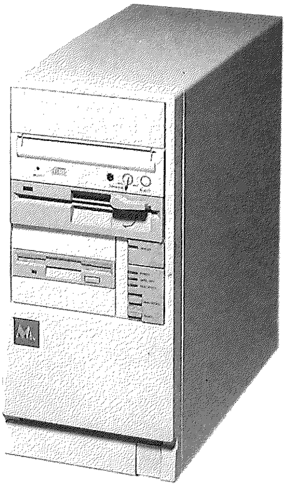
OTC 80486 MINI TOWER

¥556,000(税別、送料込み)

納期：2週間以内
保証：1年(国内保証)
CPU:80486 33MHz
CASH:256KB
RAM:4MB (1MB SIMM 70ns)
 UP TO 32MB ON BOARD
BIOS:AWORD
CHIP SET:INTEL
I/F:Serial*2/Parallel*1/SCSI*1
VIDEO:512KB SUPER VGA
(1024*768 16COLOR)
HDD:FUJITU 185MB SCSI
FDD:3.5" 1.44MB*1
KEYBOARD:101 TYPE

OPTION

4MB -> 8MB RAM	¥50,000
4MB -> 16MB RAM	¥120,000
4MB -> 32MB RAM	¥320,000
FUJITU 357MB 5" SCSI HDD	¥146,000
FUJITU 680MB 5" SCSI HDD	¥316,000
FUJITU 1.2GB 5" SCSI HDD	¥456,000
5" 1.2MB FDD	¥15,000
ATI VGA WONDER XL SUPER VGA CARD with MOUSE	¥55,000
HITACHI 14MVX 14" CRT (1024*768,800*600 Non Interlaced)	¥85,000
IBM DOS 5.0	¥32,000
IBM DOS J4.0/V	¥18,400
MS MSDOS 5.0	¥17,500



お申し込みは、住所、氏名、電話番号等を明記の上、FAXまたは封書にて承ります。折り返し、振り込み先口座、納期等を御連絡いたします。官費、校費、カードによる支払い、分割払い等はお受けつけしておりません。悪しからずご了承ください。

広告有効期限 '91／10月末日

米国The C Users Group日本支部
VILLAGE CENTER INC.
■本部：〒101 東京都千代田区三崎町3-1-16神田アメレックスビル2F
TEL 03-3239-2187 FAX.03-3221-1768
■Shop：〒101 東京都千代田区神田神保町1-35-15 たるまビル1F
TEL 03-3219-6113

スタッフ募集中!!▶明るくヤル気のある方大歓迎です。詳しくは電話3239-2187 担当:中村までお問い合わせください。

〈資料請求番号008〉

日本語Microsoft Windows 3.0環境ファイル機能一覧表

《》内は既定値
（日本IBM PS/55シリーズ/日本電気PC-9801シリーズ/セイコーエプソンPC-286/386シリーズ）

■WIN.INIファイル

●[WINDOWS]セクション

Load=＜ファイル名＞《なし》
起動時，アイコンとして自動実行したいアプリケーションの指定

Run=＜ファイル名＞《なし》
起動時，自動実行したいアプリケーションの指定

Beep=＜yesまたはno＞《yes》
警告音を出すかの指定

Spooler=＜yesまたはno＞《yes》
プリンタへの出力をプリンタマネージャを通すかの指定

Device=＜出力デバイス名＞，＜デバイスドライバ＞，＜接続ポート＞《なし》
デフォルトプリンタの定義

DeviceNotSelectedTimeout=＜秒数＞《15》
プリンタ出力タイムアウトの指定

TransmissionRetryTimeout=＜秒数＞《45》
データの再転送タイムアウトの指定

BorderWidth=＜数値＞《3》
すべてのウィンドウの境界線幅の指定（1から49）

CursorBlinkRate=＜1/1000秒単位の数値＞《530》
選択カーソル点滅間隔の指定

KeyboardSpeed=＜1/1000秒単位の数値＞《31》（98シリーズでは無効）
キーリピート間隔の指定

DoubleClickSpeed=＜1/1000秒単位の数値＞《452》
ダブルクリックとして判断する最大の間隔の指定

MouseSpeed=＜0，1または2＞《1》
マウスの移動がMouseThreshold1またはMouseThreshold2のどちらかの値を超えたときの，マウスとマウスカーソルの関係の指定（0:マウスカーソルの移動速度は変化しない／1:マウスの移動がMouseThreshold1を超えたときに，マウスカーソルは通常の2倍の速度で移動／2:マウスの移動がMouseThreshold1を超えたときに，マウスカーソルは通常の2倍の速度で移動し，MouseThreshold2を超えたときに，マウスカーソルは通常の4倍の速度で移動）

MouseThreshold1=＜ピクセル数＞《5》
マウス割り込みの間にマウスが移動できる最大のピクセル数の指定

MouseThreshold2=＜ピクセル数＞《10》
マウス割り込みの間にマウスが移動できる最大のピクセル数を指定

Programs=＜拡張子＞《com exe bat pif》
アプリケーションとして扱うファイルの指定

Documents=＜拡張子＞《なし》
ドキュメントとして扱うファイルの指定

NetWarn=＜0または1＞《1》
ネットワークの警告メッセージの表示をするかの指定（0:しない／1:する）

NullPort=＜文字列＞《"None"》
ヌルポートに付ける名前の指定

●[DESKTOP]セクション

GridGranularity=＜グリッド（8ピクセル単位）単位の数値＞《0》
画面上にウィンドウを配置する間隔の指定（0から49）

IconSpacing=＜ピクセル数＞《77（エプソンではノーマルモード時105，ハイレゾモード時184）》
アイコンの間隔の指定

Pattern=＜b1 b2 b3 b4 b5 b6 b7 b8＞《"（なし）"》
スクリーンの背景の塗りつぶしに使うパターンの指定

TileWallpaper=＜0または1＞《0》
壁紙表示方法の指定（タイル表示:0／中央に表示:1）

Wallpaper=＜ファイル名＞《"（なし）"》
壁紙に使うビットマップファイルの指定

WallpaperOriginX=＜x座標＞《0》
壁紙をタイル表示する初期オフセットの指定

WallpaperOriginY=＜y座標＞《0》
壁紙をタイル表示する初期オフセットの指定

●[EXTENSIONS]セクション

＜拡張子＞=＜コマンドライン＞
ドキュメントファイルを選択したときに実行されるコマンドラインの指定

●[INTL]セクション

iCountry=＜国別コード＞《1（エプソンでは81）》
国別コードの指定（各国の国際電話コードと同じ，ただしカナダの国別コードは2）

iCurrDigits=＜数値＞《2》
通貨表示における小数点以下の桁数の指定

iCurrency=＜数値＞《0》
通貨記号を付加する位置の指定（0:直前／1:直後／2:半角1バイト空けて前／3:半角1バイト空けて後）

iDate=＜数値＞《0（エプソンでは2）》
日付書式の指定（0:MM/DD/YY／1:DD/MM/YY／2:YY/MM/DD，2.xとの互換性を保つために用意されたもので，3.0では使用されない）

iDigits=＜数値＞《2》
数値表示における小数点以下の桁数の指定

iLZero=＜0または1＞《0（エプソンでは1）》
小数の前に0を付けるかの指定（0:つけない／1:つける，実際の小数点記号には，sDecimalで指定した値が使われる）

iMeasure=＜0または1＞《1（エプソンでは0）》
測量単位の指定（0:メートル／1:インチ）

iNegCurr=＜数値＞《0》
通貨の負の数の書式を指定

iTime=＜数値＞《0（エプソンでは1）》
時刻の表示形式の指定（0:12時間制／1:24時間制，実際の時刻の区切りには，sTimeで指定した記号が使われる）

iTLZero=＜数値＞《0》
時刻の先頭に0を付けるかの指定（0:つけない／1:つける，実際の時刻の区切りには，sTimeで指定した記号が使われる）

s1159=＜文字列＞《AM（エプソンでは「なし」）》
12時間制の時刻書式で，午前の時刻の後ろに付ける文字列の指定

s2359=＜文字列＞《PM（エプソンでは「なし」）》
12時間制の時刻書式で，午後の時刻の後ろに付ける文字列の指定

sCountry=＜文字列＞《Japan》
標準設定に使用する国の指定

sCurrency=＜文字列＞《¥》
通貨記号の指定

sDecimal=＜文字列＞《.（ピリオド）》
数値の整数部と小数部の区切りに使う記号の指定

sLanguage=＜文字列＞《usa》（98シリーズ，エプソンでは無効）
操作の基準となる言語の指定

sList=＜文字列＞《,（カンマ）》
リスト中の項目の区切りに使う文字の指定

sLongDate=＜日付書式＞《dddd, MMMM d, yyyy（98シリーズ，エプソンではyyyy MMMM dd）》
日付書式（ロング形式）の指定

aShortDate=＜日付書式＞《M/d/yy（98シリーズ，エプソンではyy/MM/dd）》
日付書式（ショート形式）の指定

sThousand=＜文字列＞《,（カンマ）》
1000単位の区切りに使う記号の指定

sTime=＜文字列＞《:（コロン）》
時刻の区切りに使う文字の指定

●[PORTS]セクション

＜ポート名＞：=＜ボーレート＞，＜パリティ＞，＜ワード長＞，＜ストップビット＞[[，p]]（エプソンではpの指定はできない）
出力ポートの指定

●[FONTS]セクション

＜フォント名＞=＜ファイル名＞
フォントファイルの指定

●[PRINTERPORTS]セクション

＜デバイス名＞=＜ドライバ名＞，＜ポート名＞，＜DeviceNotSelectedTimeout＞，＜TransmissionRetryTimeout＞[[，＜ポート名＞，＜DeviceNotSelectedTimeout＞，＜TransmissionRetryTimeout＞，...]]
アクティブまたはインアクティブなプリンタの指定

●[DEVICES]セクション

＜デバイス名＞=＜ドライバ名＞，＜ポート名＞，[[，＜ポート名＞，...]]
アクティブなプリンタの指定

●[COLORS]セクション

＜構成要素＞=＜赤＞＜緑＞＜青＞
＜赤＞，＜緑＞，＜青＞は，それぞれ赤，緑，青の明るさの指定（0～255）

■SYSTEM.INIファイル

●[BOOT]セクション

286grabber=＜ファイル名＞《なし》
MS-DOSアプリケーションを表示させるディスプレイデバイスドライバの指定

386grabber=＜ファイル名＞《なし》
エンハンストモードでMS-DOSアプリケーションを表示させるディスプレイデバイスドライバの指定

comm.drv=＜ファイル名＞《なし》
コミュニケーションドライバの指定

display.drv=＜ファイル名＞《なし》
ディスプレイドライバの指定

fixedfon.fon=＜ファイル名＞《なし》
2.xで使用する固定システムフォントファイルの指定

fonts.fon=＜ファイル名＞《なし》
3.0で使用するプロポーショナルシステムフォントファイルの指定

keyboard.drv=＜ファイル名＞《なし》
使用するキーボードドライバの指定

language.dll=＜ライブラリ名＞《なし》（98シリーズ，エプソンでは無効）
言語固有の機能を提供するDLLの指定

mouse.drv=＜ファイル名＞《なし》
マウスドライバの指定

network.drv=＜ファイル名＞《なし》
ネットワークドライバの指定

oemfonts.fon=＜ファイル名＞《なし》
OEM文字セットフォントファイルの指定

shell=＜ファイル名＞《なし》
Windowsを起動するときに実行するWindowsプログラムの指定

sound.drv=＜ファイル名＞《なし》
サウンドドライバの指定

system.drv=＜ファイル名＞《なし》
システムハードウェアドライバの指定

●[BOOT.DESCRIPTION]セクション

このセクションは，セットアップを実行するときに変更することのできるデバイス名が記述されている．これらの指定を変更してはならない．

●[KEYBOARD]セクション

keyboard.dll=＜ファイル名＞《なし》（98シリーズ，エプソンでは無効）
キーボードのレイアウトを定義するDLLの指定

oemansi.bin=＜ファイル名＞（98シリーズ，エプソンでは無効）
OEM/ANSIコードページ変換テーブル定義ファイルの指定（使用していない）

subtype=＜番号＞《なし》（98シリーズ，エプソンでは無効）
同じレイアウトのキーボードが持つ特殊機能を区別するために定義

type=＜番号＞《なし》（98シリーズ，エプソンでは無効）
利用するキーボードタイプの定義

helpkey=＜キー名＞《end》（98シリーズのみ）
キー名にhelpまたは空白を設定すると，HELPキーの，endを設定するとENDキーの仮想キーコードが出力される

clrkey=＜キー名＞《home》（98シリーズのみ）
キー名にclrまたは空白を設定するとCLRキーの，homeを設定するとHOMEキーの仮想キーコードが出力される

●[NONWINDOWSAPP]セクション

NetAsynchSwitching=＜0または1＞《0》（エプソンでは無効）
非同期ネットワークBIOSコールを検出したとき，アプリケーションの切り替えを行うかの指定（0:しない／1:する）

ScreenLines=＜番号＞《25》（98シリーズ，エプソンでは無効）
MS-DOSアプリケーションを実行するとき，表示される画面の行数を指定

SwapDisk=＜ドライブ名ーコロン（：）ーディレクトリ＞《（環境変数TEMPで指定されたディレクトリ．TEMP変数がない場合は，デフォルトでWindowsのシステムがインストールされているディレクトリ）》
MS-DOSアプリケーションのスワップ先の指定

●[STANDARD]セクション

Int28Filter=＜数値＞《10》
INT28h割り込みの割合の指定

NetHeapSize=＜キロバイト単位の数値＞《8》
ネットワーク上でデータを転送するときに基本メモリに割り当てるバッファサイズの指定

PadCodeSegments＝＜0または1＞《0》(98シリーズでは無効)
80286をステップ実行させるかの指定(0:しない／1:する)

ReservedLowMemory＝＜キロバイト単位の数値＞《0》
Windows以外のプログラムが予約する標準メモリの大きさの指定

●[386ENH]セクション

AllVMsExclusive＝＜論理値＞《false》
アプリケーションを占有フルスクリーンモードで実行するかの指定

AllKeyDelay＝＜秒単位の数値＞《.005》
キーボードの割り込みを処理する時間の指定

AllPasteDelay＝＜秒単位の数値＞《.025》
[ALT](98シリーズでは[GRPH])キーを認識する時間の指定

CGA40WOA.FON＝＜ファイル名＞《なし》(98シリーズ,エプソンでは無効)
40桁25行または25行以下のディスプレイで,MS-DOSアプリケーションを実行するときに使用される固定ピッチディスプレイフォントファイルの指定

CGA80WOA.FON＝＜ファイル名＞《なし》(98シリーズ,エプソンでは無効)
80桁25行または25行以下のディスプレイで,MS-DOSアプリケーションを実行するときに使用される固定ピッチディスプレイフォントファイルの指定

CGANoSnow＝＜論理値＞《no》(98シリーズ,エプソンでは無効)
IBM CGAディスプレイで画面がちらつかないようにするかの指定

AUXxAutoAssign＝＜秒単位の数値＞《2,98シリーズは－1,エプソンでは2》(98シリーズ,エプソンではCOMxAutoAssign)
接続されている各通信ポートのcontention検出値の指定(xはポート番号)

AUXxBase＝＜ポート＞《AUX1Base＝3F8h;AUX2Base＝2F8h;AUX3Base＝2E8h;AUX4Base＝2E0h》(98シリーズでは無効)
最初のシリアルポートアダプタの指定(xはポート番号)

AUXBoostTime＝＜1/1000秒単位の数値＞《2》(98シリーズ,エプソンではCOMBoostTime)
仮想マシンの通信割り込み時間の指定

AUXxbuffer＝＜数値＞《128》(98シリーズ,エプソンではCOMxbuffer)
通信ポートのバッファサイズの指定(xはポート番号)

AUXxIrq＝＜番号＞《AUX1Irq＝4;AUX2Irq＝3;AUX3Irq＝4;AUX4Irq＝3》(98シリーズでは無効)
通信ポートがどの割り込みラインを使うかの指定(xはポート番号)

AUXIrqSharing＝＜論理値＞《Micro Channel(TM)およびEISAを使用するときはtrue,その他のマシンを使うときはfalse》(98シリーズでは無効)
AUX IRQが,複数の通信ポート間または他のデバイスとの間で共有できるかの指定

AUXxProtocol＝＜XOFFまたは空白＞《(空白またはXOFF以外のエントリ)》(エプソンではCOMxProtocol,98シリーズでは無効)
エンハンストモードで仮想マシンからXOFF文字を受け取ったら,仮想マシンへの文字の送信を中止するかの指定(xはポート番号)

Device＝＜ファイル名または＊デバイス名＞《なし》
エンハンストモードで使用する仮想デバイスの指定

Display＝＜ファイル名または＊デバイス名＞《なし》
エンハンストモードで使用するディスプレイデバイスの指定

GRAB386.FON＝＜ファイル名＞《なし》(98シリーズのみ)
非Windowsアプリケーションで使用される固定ピッチディスプレイフォントファイルの指定

DMABufferIn1MB＝＜論理値＞《no》(98シリーズ,エプソンでは無効)
DMAバッファメモリを最初の1MBに割り当てるかの指定

DMABufferSize＝＜キロバイト単位の数値＞《16》(98シリーズ,エプソンでは無効)
DMAバッファとして予約するメモリ容量の指定

DualDisplay＝＜論理値＞(98シリーズ,エプソンでは無効)
B000:0000からB7FF:000Fのメモリをディスプレイアダプタ用に確保するかの指定

Crt16WOA.fon＝＜ファイル名＞《CRT16WOA.FON》(エプソンのみ)
ノーマルモードでMS-DOSアプリケーションを実行するとき使う固定ピッチディスプレイフォントファイルの指定

Crt24WOA.fon＝＜ファイル名＞《CRT24WOA.FON》(エプソンのみ)
ハイレゾモードでMS-DOSアプリケーションを実行するとき使う固定ピッチディスプレイフォントファイルの指定

EBIOS＝＜ファイル名または＊デバイス名＞《なし》
エンハンストモードで使用される増設BIOSデバイスの指定

EMMExclude＝＜パラグラフ－範囲＞《なし》
未使用のアドレス領域を探すときにスキャンしないメモリの範囲の指定

EMMInclude＝＜パラグラフ－範囲＞《なし》
未使用のアドレス領域を探すときにスキャンするメモリの範囲の指定

EMMPageFrame＝＜パラグラフ＞《なし》
適切なページフレームが見つからないときに,64KBのページフレームが始まるパラグラフの指定

LastEMMSeg＝＜パラグラフ＞《なし》(98シリーズのみ)
エンハンストモードのWindowsのページフレーム終了アドレスを指定

HDBIOSMove＝＜論理値＞《off》(98シリーズのみ)
SCSIおよび従来インターフェース固定ディスクのROM-BIOSの位置を移動

VM1LargePage＝＜論理値＞《off》(98シリーズのみ)
ラージページEMSを使用するかの指定

EMMSize＝＜キロバイト単位の数値＞《65,536》(98シリーズでは65,535)
EMSメモリとしてマップ可能なメモリ容量の指定

FileSysChange＝＜論理値＞《ON,エプソンではOFF》
MS-DOSアプリケーションが出すメッセージを,ファイルマネージャが受け取るかの指定

Global＝＜デバイス名＞《(すべてのデバイス)》
CONFIG.SYSファイルでロードされるグローバルなDOSデバイスの定義

HighFloppyReads＝＜論理値＞《yes》(98シリーズでは無効)
DMAのベリファイでE000:0000からEFFF:000Fの領域を利用するかの指定

IgnoreInstalldEMM＝＜論理値＞《no》
他社EMMを実行しているときでも,エンハンスドモードで起動するかの指定

InDosPolling＝＜論理値＞《no》
メモリ常駐ソフトがInt21Hフックのオペレーションを許可するかの指定

INI28Critical＝＜論理値＞《true》
メモリ常駐ソフトで使用されるINT28hを処理するクリティカルセクションが必要であるかの指定

INT1ACritical＝＜論理値＞《false》(98シリーズのみ)
常駐プログラムが使用するINT1Ah割り込みを処理するために,クリティカルセクションが必要かどうかを指定

IRQ9Global＝＜論理値＞《no》(98シリーズ,エプソンでは無効)
IRQ9マスクをグローバルに変換するかの指定

Keyboard＝＜ファイル名または＊デバイス名＞《none》
エンハンストモードで使用するキーボードドライバの指定

KeyBoostTime＝＜秒単位の数値＞《.001》
キー入力を受け取ったとき,優先的に実行するアプリケーションに与える時間の指定

KeyBufferDelay＝＜秒単位の数値＞《.2》
キーボードバッファがフルのとき,キーボード入力のペーストを遅らせる時間の指定

KeyIdleDelay＝＜秒単位の数値＞《.5》
仮想マシンへのキー入力をシミュレートしたあとで,アイドルコールを無視する時間の指定

KeyPasteDelay＝＜秒単位の数値＞《.003》
キーをペーストしたあと次の文字をペーストするまでの待ち時間の指定

KeyPasteTimeout＝＜秒単位の数値＞《1》(98シリーズ,エプソンでは無効)
キーボード入力の読み込みに必要なBIOSコールをする時間の指定

Local＝＜デバイス名＞《none》
各仮想マシンに対してローカルでなければならないMS-DOSデバイスドライバの定義

LPTxAutoAssign＝＜秒単位の数値＞《60》
接続した各ポートのcontention検出値の指定(xはポート番号)

MapPhysAddress＝＜範囲＞《none》
メモリマネージャが前もって物理ページテーブルのエントリと連続アドレス領域に割り当てるアドレス範囲の指定

MaxPagingFileSize＝＜キロバイト単位の数値＞《none》
テンポラリスワップファイルの最大値の指定

MinTimeSlice＝＜1/1000秒単位の数値＞《20》
仮想マシンが他の仮想マシンに切り替わるまでに,実行できる最小時間の指定

MinUserDiskSpace＝＜キロバイト単位の数値＞《500》
テンポラリスワップファイルを作成するときに,使用可能なディスク領域をどのくらい残すかの指定

Mouse＝＜ファイル名または＊デバイス名＞《none》
エンハンストモードでマウスハードウェアを制御する仮想デバイスの指定

MouseLrg＝＜番号＞《13》(エプソンのみ)
マウスのハードウェア割り込みラインの指定

MouseSoftInit＝＜論理値＞《true》(エプソンでは無効)
INT33hファンクション0のハードイニシャライゼーションコールを,マウスハードウェアをリセットしないファンクション33ソフトイニシャライゼーションコールに変換するかの指定

NetAsynchFallback＝＜論理値＞《false》
ネットワークBIOSリクエストの失敗をセーブするように知らせるかの指定

NetAsynchTimeout＝＜秒単位の数値＞《5.0》
非同期ネットワークBIOSのリクエストに応えるために,クリティカルセクションに入る必要がある場合の,タイムアウトの時間の指定

NetDMASize＝＜キロバイト単位の数値＞《32(マイクロチャネル(TM)使用のマシン),0(マイクロチャネル(TM)を使用しないマシン)(98シリーズでは無効)》
ネットワークがインストールされているとき,ネットワークBIOS転送ソフトが使用するDMAバッファサイズの指定

NetHeapSize＝＜キロバイト単位の数値＞《12》
エンハンストモードで基本メモリに割り当てるネットワーク上でデータを転送するときに使うバッファのサイズの指定

Network＝＜ファイル名または＊デバイス名＞《none》
エンハンストモードで使用するネットワークの種類の指定

NMIReboot＝＜論理値＞《no》(98シリーズ,エプソンのみ)
ノンマスカラブルな割り込みを受け取ったときリブートさせるかの指定

NoEMMDriver＝＜論理値＞《false》
エンハンストモードでEMSメモリドライバをインストールしないかの指定

Paging＝＜論理値＞《yes》
デマンドページング(仮想メモリ)を有効にするかの指定

PagingDrive＝＜ドライブ名＞《none》
エンハンストモードでテンポラリスワップファイルを割り当てるディスクドライブの指定

PerVMFILES＝＜数値＞《10》
各仮想マシンに割り当てるプライベートファイルハンドル数の指定

PSPIncrement＝＜数値＞《2》
UniqueDOSPSPを有効にするとき,連続する各仮想マシンに予約する追加メモリ量の指定

ReflectDosInt2A＝＜論理値＞《false》(98シリーズではReflectDosInt2Ah)
DOS INT 2AHシグナルを有効にするかの指定

ReservePageFrame＝＜論理値＞《true》
DOS転送バッファをEMSページフレーム領域かまたは基本メモリのどちらかに割り当てなければならないとき,どちらを使うかの指定

BASICROMUse＝＜論理値＞《ON》(98シリーズのみ)
N88-日本語BASIC(86)のROM空間を使用するかの指定

SGrabLPT＝＜ポート番号＞《none》(エプソンでは無効)
指定したポートのプリンタ割り込みを,現在の仮想マシンではなく,システムの仮想マシンへ渡します.

SystemROMBreakPoint＝＜論理値＞《true》
ブレークポイントとしてF000:000から1MBの範囲でROMアドレス領域を使用するかの指定

SysVMEMSLimit＝＜キロバイト単位の数値＞《2048》
Windowsが利用できるEMSメモリの容量の指定

SysVMEMSLocked＝＜論理値＞《no》
EMSメモリをハードディスクにスワップするかの指定

SysVMEMSRequired＝＜キロバイト単位の数値＞《0》
Windowsを起動するために必要なEMSメモリ容量の指定

SysVMV86Locked＝＜論理値＞《false》
システムの仮想マシンで使われている仮想モードメモリを,ディスクにスワップアウトしないで,メモリ中にロックされたままにするかの指定

SysVMXMSLimit＝＜キロバイト単位の数値＞《2048》
XMSメモリドライバが,DOSデバイスドライバやシステム仮想マシンのメモリ常駐ソフトに割り当てる,最大メモリ容量の指定(-1を指定するとアプリケーションが要求する使用可能なすべてのXMSメモリを割り当てる)

SysVMXMSLocked＝＜論理値＞《no》
XMSメモリドライバによって割り当てられたメモリをハードディスクにスワップするかの指定

SysVMXMSRequired＝＜キロバイト単位の数値＞《0》
Windowsを起動するために必要な,XMSドライバで予約するXMSメモリ容量の指定

TimeCriticalSection＝＜1/1000秒単位の数値＞《0》(98シリーズ,エプソンではTimerCriticalSection)
すべてのタイマ割り込みコードの前後のクリティカルセクションに入り込むよう指示し,タイムアウト時間の指定

TranslateScans＝＜論理値＞《no》(98シリーズ,エプソンでは無効)
キーボードスキャンコードを,IBM標準コードと互換性のあるコードに変換するかの指定

UniqueDOSPSP＝＜論理値＞《false》
すべてのアプリケーションは個別のアドレスで起動するかの指定

UseInstFile＝＜論理値＞《false》(98シリーズのみ)
DOS内のデータ構造がローカルでなければならないかどうかを決定するのに使うことができる情報を,INSTANCE.386ファイルの中から探すかの指定

UseWarning＝＜論理値＞《true》(98シリーズではVCPIWarning)
アプリケーションがVCPIを使おうとするとき,警告メッセージを表示するかの指定

VirtualHDIrq＝＜論理値＞《on》(98シリーズ,エプソンでは無効)
割り込みを処理するROMルーチンをバイパスして,ハードディスクコントローラの割り込みを終了するかの指定

WindowKBRequired＝＜キロバイト単位の数値＞《256》
Windowsを起動するために必要な基本メモリ量の指定

WindowMemSize＝＜-1またはキロバイト単位の数値＞《-1》
Windows自身で使用する基本メモリの容量の指定(-1～640,-1を指定するとWindowsが必要な分だけこの領域を使用)

WindowUpdateTime＝＜1/1000秒単位の数値＞《50》
MS-DOSアプリケーションをウィンドウ内で再描画するときの時間の指定

WinExclusive＝＜論理値＞《no》
Windowsアプリケーションがフォアグラウンドにあるとき,つねにコンピュータのすべての処理時間を使うかの指定

WinTimeSlice＝＜数値,数値＞《100,50》
1番目の数値は,バックグラウンドで実行しているMS-DOSアプリケーションに対して,フォアグランドで実行しているWindowsアプリケーションすべてに与えられる相対的な処理時間(MinTimeSliceで指定した時間)の指定,2番目の数値は,非Windowsアプリケーションがフォアグラウンドで実行しているときに,バックグラウンドで実行しているすべてのWindowsアプリケーションに与えられる相対的な処理時間の指定

8088から80486までの全命令に対応!!

コメント付きのアセンブラソースを作成する世界で最も優れた逆アセンブラ

SOURCER

全てのMS-DOSプログラマーに捧げる限界を越えたコード解析ツール

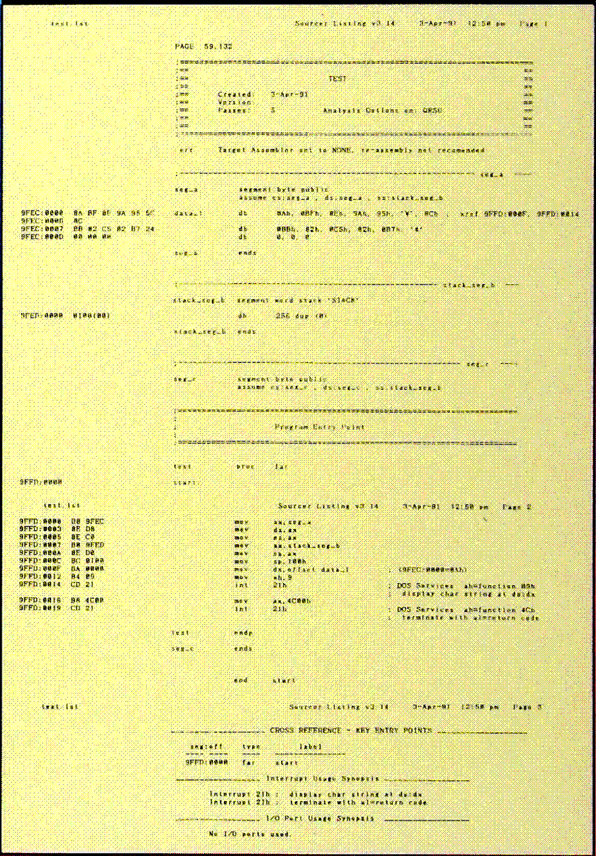
SOURCERはアセンブラやC, FORTRANなど、あらゆる言語によって作成された、様々なMS-DOS実行形式ファイルから、処理内容に対応した適切なコメントを作った分かりやすいアセンブラ・ソースリストを作成する、リバース・エンジニアリングに不可欠なコード解析ツールです。

解析の正確さ、出力ソース・コードの分かりやすさ、さらに解析可能な命令セットの範囲の広さ、操作性の良さなど、SOURCERはあらゆる面で他に類を見ない優れた逆アセンブラです。

こんな時に威力を発揮します

- ■ドキュメントのないソースコードの解析を行いたい
- ■ソースが入手不可能なプログラム(市販の各種ソフトウエア)に対して修正を行いたい
- ■プログラム(市販の各種ソフトウエア等)が実際どのように動作しているかを知りたい
- ■ソース・コードを紛失してしまった
- ■「ウイルス」を含んでいる恐れのあるプログラムをソースコード・レベルで検査したい

解析逆アセンブルリスト出力サンプル



SOURCERの特長

●以下の命令セット実行プログラムの解析逆アセンブルが可能です。

8086　87　8088　80186　188　80286　287
80386　387　80486　V20　30

●以下の種類の実行形式プログラムに対して、ユーザが指定する任意のアドレス範囲に対して逆アセンブルが可能です。

- ▶COMファイル　▶EXEファイル　▶デバイス・ドライバ
- ▶オーバーレイ・ファイル　▶SYSやBINなど、その他のバイナリ・ファイル
- ▶メモリ上に既にロケーションされている(ROM領域も含めて)プログラム

●複雑に入り混じるマルチ・セグメント構成に対して、最大250個までのセグメントの解析が可能です。

●各種アセンブラ処理系に特定対応した逆アセンブラ・ソース・リストの作成もできます。

MASM-4.0　MASM-5.0　MASM-5.1
TASM-1.0　TASM-2.0　OPTASM　その他

●解析結果の逆アセンブル・リストには、使用されているファンクション・コールの機能と各レジスタの意味、アクセスしているデータ領域に関する細かい情報、I/Oポートの操作状況、複雑なコードの処理内容などについての、詳細なドキュメントに類するコメントが付加されます。

●解析オプションを指定すると、解析方法を、ユーザ側で細かく設定できます。さらに、定義ファイルを使用することによって、逆アセンブル・ソース・リストのカスタマイズも行えます。

●最大9パスまでの綿密な解析逆アセンブリ処理を行います。

価格:¥38,000

【動作機種】 ▶PC-9800シリーズ ▶FMRシリーズ ▶J3100シリーズ ▶Panacom Mシリーズ ▶PS/55シリーズ ▶AX

【動作環境】 日本語MS-DOS Ver. 2/3/4, DOS/V J4.0
必要メモリ:512KB以上

【供給メディア】 5インチまたは3.5インチ フロッピー・ディスク

完全日本語マニュアルを装備

●期間限定トレードイン特別販売のお知らせ●

- ■現在ご使用になっている輸入版SOURCERや他の逆アセンブラを特別価格で日本版SOURCERにお取り替え致します。
- ■価格や申し込み方法等、詳細についてはThe Programmer's INDEX (TEL:03-3293-3887)までお問い合わせ下さい。
- ■トレードイン実施期間:'91年6月1日〜10月31日迄

【お求め方法】当社製品は、パソコンソフト取扱い店でお求めいただけますが、弊社に直接ご注文の際は、The Programmer's INDEX (TEL:03-3293-3887)までお問い合わせ下さい。※各製品の表示価格には、消費税は含まれておりません。

株式会社 LIFEBOAT
〒101 東京都千代田区神田錦町3-6
PHONE:03-3293-4711代 FAX:03-3293-4710

●Copyright Notice…… ＊SOURCERはV Communications Inc.の商標です ＊MS-DOSはMicrosoft Corp.の登録商標です ＊その他、プログラム名、システム名、CPU名は一般に各メーカの登録・商標です ＊価格・仕様は予告なく変更する場合があります

〈資料請求番号P03〉

MITSUBISHI

技術がつくる高度なふれあい *SOCIO-TECH*

［32bitでノート形、だからビジネスがスピーディ。］

「楽勝か、惜敗か。」は判断の速さで決まる。「楽勝研究会」の選ぶパソコンは「MAXYNOTE386」。32bit CPU搭載で、さまざまな情報を高速処理。交換自在のハードディスクパック方式で、大容量情報を自由自在に操れる。だから、ビジネスがスピーディだ。コンピュータと通信の高度な技術でニーズにこたえる、三菱情報通信システム。ビジネス「楽勝」のヒケツです。

- ●32bitCPU386™SX搭載。ビジネスを高速処理。
- ●ワンタッチで交換自在、高速&大容量ハードディスク内蔵。
- ●多彩なシステムアップ能力、MAXYSTATIONⅡ。

三菱AXパソコン
MAXYNOTE386

■標準価格…398,000円(税別：HDD 20MBモデル)

スピードの遅いやつは、国際ビジネスについてけない。

Hiroyuki Sanada

ニーズから発想する三菱情報通信システム
Think&Link

AX

386™SXは米国インテル社、MAXY、MAXYSTATIONは三菱電機㈱の登録商標です。
MAXYメニューシステムは、㈱エージーソフトのAZ-MOS(エージーモス)をベースとしたメニューシステムです。AZ-MOSは㈱エージーソフト㈱アイシーエムが開発したインストール方式「IBFファイル(インストールバッチファイル)システム」を採用しています。
世界標準化を推進するAX。日本のソフトウェアメーカ、ハードウェアメーカ、流通各社によるAX協議会★のもと、OS、キーボード、画面、プリンタ部を規格化。国内・海外の豊富なソフトや周辺機器の共通利用を実現します。★1991年7月末現在参加約607社

三菱電機株式会社

お問合せは…三菱電機株式会社 〒100 東京都千代田区丸の内2-2-3(三菱電機ビル) 本社情報処理機器事業部計画部(03)3218-3285 本社情報処理機器部(03)3218-3515 本社システム営業第一部(03)3218-3547 本社システム営業第二部(03)3218-3298 本社情報通信システム計画部(03)3218-3236 本社電機事業部(03)3218-3158 ●札幌(011)212-3739 ●仙台(022)264-5656 ●大宮(048)653-0261 ●千葉(0472)41-8671 ●横浜(045)212-3521 ●新潟(025)241-7220 ●金沢(0762)52-8821 ●名古屋(052)565-3246 ●大阪(06)347-2421 ●広島(082)248-5330 ●高松(0878)25-0009 ●福岡(092)721-2346

カタログ請求券 AXパソコン プログラマーズ 9111

T4911777711982 〈資料請求番号P04〉 雑誌コード17777-11